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 extensible by multiple applications requesting different kinds of protected objects.

- a variety of security mechanisms based on public or secret key cryptography
- 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-00.txt.

2. The GAA API data types and calling conventions

The data types describe only fields that must be provided by all applications. Individual implementations may provide additional fields for internal use within GAA API routines.

2.1. Integer types

The GAA API defines the following integer data type:

```c
uint32   32-bit unsigned integer
```

2.2. String and similar data

Certain data items used by the GAA API may be regarded as character strings, e.g. principal names or specification of security mechanisms. The data of this kind is passed between the GAA API and application using the gaa_data data type, which is a pointer to a gaa_data structure.

2.2.1. gaa_data data structure

The gaa_data type is a structure containing the following fields:

```c
typedef struct gaa_data_struct gaa_data, *gaa_data_ptr;

struct gaa_data_struct {
    int length;
    char *data;
};
```

2.2.2. gaa_tag_value_list data structure

Some data items used by the GAA API may be regarded as a linked list of tag:vector elements. For example, a pointer to a list of tag:vector elements is passed to the gaa_check_authorization routine via the gaa_options parameter.

The data of this kind is passed between the GAA API and application using the gaa_data data type, which is a pointer to a gaa_data structure.

```c
typedef struct gaa_tag_value_list_struct  gaa_tag_value_list, *gaa_tag_value_list_ptr;

struct gaa_tag_value_list_struct {
    gaa_data_ptr             tag;
    gaa_data_ptr             value;
    gaa_tag_value_list_ptr   next;
};
```

2.2.3. gaa_tag_vector and gaa_tag_vector_list data structures

Some data items used by the GAA API may be regarded as a linked list of tag:vector elements. For example, a pointer to a list of tag:vector elements is passed to the gaa_check_authorization routine via the gaa_options parameter.

The data of this kind is passed between the GAA API and application using the gaa_data data type, which is a pointer to a gaa_data structure.

```c
typedef struct gaa_tag_vector_struct  gaa_tag_vector, *gaa_tag_vector_ptr;

struct gaa_tag_vector_struct {
    gaa_data_ptr   tag;
    uint32         vector;
};
```
The `gaa_tag_vector_list` type is a structure containing the following fields:

- **element**: Contains a pointer to the element of type `gaa_tag_vector`
- **next**: Contains a pointer to the next element in the list

```c
typedef struct gaa_tag_vector_list_struct  gaa_tag_vector_list,
    *gaa_tag_vector_list_ptr;
```

```c
struct gaa_tag_vector_list_struct {
    gaa_tag_vector_ptr       element;
    gaa_tag_vector_list_ptr  next;
};
```

### 2.3 Opaque data types

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

Opaque data is passed between the GAA API and the application using the `gaa_buffer_ptr` data type, which is a pointer to a `gaa_buffer` structure.

The `gaa_buffer` type is a structure containing the following fields:

- **length**: Contains the total number of bytes in the datum
- **value**: Contains a pointer to the actual datum

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

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

### 2.4 GAA API Access Control List data structures

Certain GAA API procedures take parameters of the type `gaa_acl_ptr`, which is a pointer to a `gaa_acl` structure.

The `gaa_acl` data type contains a complete ACL, made up of a list of entry fields of type `gaa_acl_entry`. `GAA_ACL_UNBOUND` defines the value of the unknown ACL handle.

#### 2.4.1 gaa_acl data structure

The `gaa_acl` type is a structure containing the following fields:

- **object_name**: Contains a name of an object with which the ACL is associated. The name of the object is from the application-specific name space. The `object_name` field is evaluated by GAA API routine when checking credentials of the type `gaa_authorized_cred` to determine if the object_name for which the ACL is being examined is listed in the set of objects which can be accessed by the grantee. Whether this field should be present in the `gaa_acl` data structure is not clear. If we decide not to include this field, we should provide a means for deciding if the targeted object is included in the list of objects accessible by the grantee.

- **acl_entries**: A pointer to a list of ACL entries, each of type `gaa_acl_entry`

```c
typedef struct gaa_acl_struct gaa_acl, *gaa_acl_ptr;
```

#### 2.4.2 gaa_acl_entry data structure

The `gaa_acl_entry` type is a structure containing the following fields:

- **type**: Indicates whether the ACL entry grants or denies access. If `type` = `GAA_GRANT`, the entry specifies positive rights. If `type` = `GAA_DENY`, the entry specifies negative rights. Entries of this type must not have any conditions, so `GAA_NO_CONDITIONS` value is stored in the condition field.

- **access_rights**: A pointer to a linked list of structures of the type `gaa_acl_access_rights_set`. Each set indicates access rights named with the same tag.

- **conditions**: A pointer to a list of structures of the type `gaa_conditions`, which contains all different conditions that are present in the ACL entry.

```c
typedef enum {
    GAA_GRANT,
    GAA_DENY
} gaa_entry_type;
```

```c
typedef struct gaa_acl_entry_struct  gaa_acl_entry,
    *gaa_acl_entry_ptr;
```

```c
struct gaa_acl_entry_struct {
    gaa_entry_type             type;
    gaa_principal_list_ptr     principals;
    gaa_access_rights_set_ptr  access_rights;
    gaa_conditions_ptr         conditions;
    gaa_data_ptr            object_name;
    gaa_acl_entry_ptr       acl_entries;
    gaa_data_ptr            acl_pos;
};
```
gaa_acl_entry_ptr next;
}

2.4.3 gaa_principal and gaa_principal_list data structures

The gaa_principal type is a structure containing the following fields:

- **type**
  Specifies the type of principal: GAA_USER, GAA_GROUP, GAA_HOST, GAA_APPLICATION or GAA_ANYBODY.

- **mech_type**
  Contains principal naming method, which specifies the type of the security mechanism used to obtain authenticated subject’s identity, e.g. Kerberos.V5, DCE.

- **name**
  Contains principal name, which specifies authenticated subject’s identity, e.g. joe@ISI.EDU.

```c
typedef enum  {
    GAA_USER        ,
    GAA_GROUP       ,
    GAA_HOST        ,
    GAA_APPLICATION ,
    GAA_ANYBODY
} gaa_principal_type;
```

```c
typedef struct  gaa_principal_struct   gaa_principal, *
               gaa_principal_ptr;
```

```c
struct  gaa_principal_struct {
    gaa_principal_type    type;
    gaa_data_ptr          mech_type;
    gaa_data_ptr          name;
};
```

The gaa_principal_list type is a structure containing the following fields:

- **principal**
  A pointer to the gaa_principal structure

- **next**
  A pointer to the next principal in the ACL entry

```c
typedef struct  gaa_principal_list_struct  gaa_principal_list, *
               gaa_principal_list_ptr;
```

```c
struct gaa_principal_list_struct {
    gaa_principal_ptr     principal;
    gaa_principal_ptr     next;
};
```

2.4.4 gaa_access_rights_set data structure

The gaa_access_rights_set type is a structure containing the following fields:

- **rights**
  A pointer to a structure of the type gaa_tag_vector. In the example of an ACL entry given in the section 2.4.2, the gaa_tag_vector structure, corresponding to the first access rights set, will have “FILE” value in the “tag” field and 1110...0 in the “vector” field.

- **mask**
  An array containing GAA_NUM_ACCESS_RIGHTS pointers to a list of structures of type gaa_condition_mask. Each gaa_condition_mask structure contains a pointer to particular condition associated with the granted right and a pointer to the next structure of the type gaa_condition_mask.

In the example of an ACL entry given in the section 2.4.2, lets consider mask structure for the first access rights set. The access rights set will have “FILE” value in the “tag” field and 1110...0 in the “vector” field of “rights” element. mask[0] (corresponding to “read” access right) will be set to GAA_NO_CONDITIONS, since “read” has no conditions. mask[1] (corresponding to “write” access right) will point to the list of gaa_condition_mask, which has two elements. The first element will have a pointer to the condition time_window : 7AM-7PM in the “condition” field, the second element will have a pointer to the condition time_day : Mon-Fri in the “condition” field.

- **next**
  A pointer to the next set of access rights in the ACL entry.

```c
typedef  struct  gaa_condition_mask_struct  gaa_condition_mask, *
               gaa_condition_mask_ptr;
```

```c
struct  gaa_condition_mask_struct {
    gaa_conditions_ptr      condition;
    gaa_condition_mask_ptr  next;
};
```

2.4.5 gaa_conditions data structure

This structure defines a specific policies under which access rights are granted. In this structure all conditions that have been encountered in the ACL entry are listed. The mapping between access rights and corresponding condition sets is defined via gaa_condition_mask structure.

The gaa_conditions type is a structure containing the following fields:

- **type**
  Type defines the type of condition and which evaluation function will be invoked to evaluate it.

- **value**
  Value represents a set of parameters for the condition evaluation function. The meaning of the parameters and their relationships depend on how the evaluation function interprets them.

- **flags**
Flags indicate if condition was evaluated/not evaluated, satisfied/not satisfied, or must be enforced.

typedef struct gaa_conditions_struct gaa_conditions,
    *gaa_conditions_ptr;
struct gaa_conditions_struct {
    gaa_data_ptr        type;  
    gaa_data_ptr        value;  
    uint32              flags;  
    gaa_conditions_ptr  next;    
};

2.5 GAA API Security Context data structures

The security context is a GAA API data structure, which is passed as an argument to the GAA API. It stores information relevant to access control.

2.5.1 gaa_sec_context data structure

The gaa_sec_context type is a structure containing the following fields:

- identity_cred
  A pointer to a list of structures of the type gaa_identity_cred
- authr_cred
  A pointer to a list of structures of the type gaa_authrized_cred
- group_membership
  A pointer to a list of structures of the type gaa_identity_cred, which specifies that the grantee is a member of only listed groups
- group_non_membership
  A pointer to a list of structures of the type gaa_identity_cred, which specifies that the grantee is NOT a member of the listed groups
- attributes
  A pointer to a list of structures of the type gaa_attributes, which contains miscellaneous attributes attached to the grantee, e.g., age of the grantee, grantee security clearance
- unevl_cred
  A pointer to a list of structures of type gaa_unevaluated_cred
- connection_state
  Contains a mechanism-specific representation of per-connection context, some of the data stored here include keyblocks, address...

2.5.2 gaa_identity_cred data structure

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

typedef struct gaa_identity_cred_struct  gaa_identity_cred,
    *gaa_identity_cred_ptr;
struct gaa_identity_cred_struct {
    gaa_principal_ptr        principal;  
    gaa_conditions_ptr       conditions;  
    gaa_buffer_ptr           mech_spec_cred;  
    gaa_identity_cred_ptr    next;    
};

2.5.3 gaa_authorized_cred data structure

This type of credentials used when individuals grant delegated credential or generate a capability.

- grantor
  Specifies a principal who issued the credential
- grantee
  Specifies a principal for whom the credential was issued
- objects
  A pointer to a linked list of structures of the type gaa_data, which contains a list of objects which may be accessed by the grantee.
  Object names are from the application-specific name space.
- access_rights
  A pointer to a list of structures of the type gaa_acl_access_rights_set. Each structure indicates granted access permissions.
typedef struct gaa_authorized_cred_struct  gaa_authorized_cred,
    *gaa_authorized_cred_ptr;

struct gaa_authorized_cred_struct{
    gaa_principal_ptr         grantor;
    gaa_principal_ptr         grantee;
    gaa_data_list_ptr         objects;
    gaa_access_rights_set_ptr access_rights;
    gaa_conditions_ptr        conditions;
    gaa_buffer_ptr            mech_spec_cred;
    gaa_authorized_cred_ptr   next;
};

2.5.4  gaa_attributes data structure

The gaa_attributes type is a structure containing the following fields:

mech_type
    Security mechanism used to obtain the credential

type
    Type is used to define the type of attribute

value
    Represents actual attribute contents

conditions
    A pointer to a list of structures of the type gaa_conditions,
    which lists restrictions placed on the attribute credentials

mech_spec_cred
    Contains a handle to the actual mechanism-specific attribute
    credential

next
    Contains a pointer to the next attributes credential belonging to
    the same grantee.

typedef struct gaa_attributes_struct  gaa_attributes,
    *gaa_attributes_ptr;

struct gaa_attributes_struct {
    gaa_data_ptr          mech_type;
    gaa_data_ptr          type;
    gaa_data_ptr          value;
    gaa_conditions_ptr    conditions;
    gaa_buffer_ptr        mech_spec_cred;
    gaa_attributes_ptr    next;
};

2.5.5  gaa_unevaluated_cred data structure

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

The gaa_unevaluated_cred type is a structure containing the following fields:

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

grantor
    Specifies a principal who issued the credential

grantee
    Specifies a principal for whom the credential was issued

mech_type
    Specifies security mechanism used to obtain the credential

mech_spec_cred
    Contains a handle to the actual mechanism-specific authorization
    credential

credential_verification
    This pointer to the credential verification function for upcall is
    added by the application or transport

next
    Contains a pointer to the next unevaluated credential belonging to
    the same subject.

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

typedef struct gaa_unevaluated_cred_struct   gaa_unevaluated_cred,
    *gaa_unevaluated_cred;

struct gaa_unevaluated_cred_struct {
    gaa_cred_type              cred_type;
    gaa_principal_ptr          grantor;
    gaa_principal_ptr          grantee;
    gaa_buffer_ptr             mech_spec_cred;
    void (*cred_verification)(gaa_sec_context_ptr, va_list ap);
    gaa_unevaluated_cred_ptr   next;
};

2.6  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 type is a structure containing the following fields:

valid_time
    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.
    Expiration time is calculated by the GAA API, based on:
    - time-related conditions in the ACL matching entries
    - restrictions in the attributes, identity, authorization and
    group membership credentials

granted_raccess_rights
    A pointer to a linked list of structures of the type
    gaa_acl_access_rights_set

conditions
A pointer to a list of structures of type gaa_conditions, which lists all different conditions for the granted access rights.

Contains a pointer to a gaa_value_type_list structure, which contains information about additional security attributes required, e.g. group membership or authorized credentials.

typedef struct gaa_answers_struct gaa_answers, *gaa_answers_ptr;

typedef struct gaa_answers_struct {
    gaa_time_period_ptr        valid_time;
    gaa_access_rights_set_ptr  granted_access_rights;
    gaa_conditions_ptr         conditions;
    gaa_tag_value_list_ptr     required_sec_attributes
} gaa_answers_struct;

3. Memory allocation

Storage for data returned to the application by a GAA API function using gaa_buffer_ptr, gaa_tag_value_list_ptr, gaa_tag_vector_list is allocated by the GAA API routines. The application may clear this storage by invoking the gaa_release_buffer, gaa_release_tag_value_list and gaa_release_tag_vector_list routines.

Allocation of the gaa_buffer, gaa_tag_value_list, gaa_tag_vector_list and gaa_sec_context structures is always the responsibility of the application.

4. Status codes

One or two status codes are returned by each GAA API routine. Two distinct sorts of status codes are returned. These are the GAA API status codes and mechanism-specific status codes.

4.1 The GAA API status codes

System/Brand: Expires February 1995

GAA API routines return GAA API status codes as their gaa_error_code function value. These codes indicate errors that are independent of the underlying mechanism. The errors that can be indicated are:

- GAA_FAILURE - Failure, see minor_status for more information
- GAA_OK - Success
- GAA_PENDING - Pending
- GAA_RETRY - Retry

4.2 Mechanism-specific status codes

GAA API routines return a minor_status parameter, which is used to indicate specialized errors from the underlying mechanisms to provide additional information about GAA API errors. The GAA status code GAA_FAILURE is used to indicate that the underlying mechanism detected an error for which no specific GAA status code is defined. The mechanism status code will provide more details about the error.

5. GAA API routine descriptions

This section lists the functions performed by each of the GAA API routines and discusses their major parameters, describing how they are to be passed to the routines.

5.1 gaa_get_object_acl routine

Purpose:
The gaa_get_object_acl function is called to obtain a handle to an object ACL.

Parameters:
- minor_status - mechanism-specific status code
- 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.
- authr_db - Pointer to an application-specific authorization database
- retrieve - An upcall function for the retrieval of the object ACL.

The application maintains authorization information in a form understood by the application. It can be stored in a file, database, directory service or some other way. The upcall function provided for the GAA API retrieves the ACL and translates it into the internal GAA API representation.

acl_handle - A pointer to a handle bound to the ACL that is the subject of examination.

Function value:
- GAA_OK - Success
- GAA_FAILURE - Failure, see minor_status for more information

5.2 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:
- minor_status - mechanism-specific status code
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- sec_context - Principal's security context
- check_access_rights - List of access rights for authorization. This argument is optional.
- detailed_answer - Contains various information for further evaluation by the application.

5.3 gaa_query_authorizations routine

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

Parameters:
- minor_status - Mechanism-specific status code
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- sec_context - Principal's security context
- check_access_rights - List of access rights for authorization. This argument is optional.
- detailed_answer - Contains various information for further evaluation by the application.

5.4 gaa_release_context routine

Purpose:
The gaa_release_context function releases the context bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- sec_context - Principal's security context

5.5 gaa_release_resource routine

Purpose:
The gaa_release_resource function releases the resource bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- resource - Resource to be released, e.g. file, directory, etc.

5.6 gaa_release_object_access_rights routine

Purpose:
The gaa_release_object_access_rights function releases the object access rights bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- object - Object to be released, e.g. file, directory, etc.

5.7 gaa_release_object_access_rights_set routine

Purpose:
The gaa_release_object_access_rights_set function releases the object access rights set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- object_access_rights_set - Object access rights set to be released, e.g. file, directory, etc.

5.8 gaa_release_conditions routine

Purpose:
The gaa_release_conditions function releases the conditions bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- conditions - Conditions to be released, e.g. file, directory, etc.

5.9 gaa_release_required_sec_attributes routine

Purpose:
The gaa_release_required_sec_attributes function releases the required security attributes bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- required_sec_attributes - Required security attributes to be released, e.g. file, directory, etc.

5.10 gaa_release_access_rights_set routine

Purpose:
The gaa_release_access_rights_set function releases the access rights set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_rights_set - Access rights set to be released, e.g. file, directory, etc.

5.11 gaa_release_access_rights routine

Purpose:
The gaa_release_access_rights function releases the access right bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_right - Access right to be released, e.g. file, directory, etc.

5.12 gaa_release_time_period routine

Purpose:
The gaa_release_time_period function releases the time period bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- time_period - Time period to be released, e.g. file, directory, etc.

5.13 gaa_release_access control_list routine

Purpose:
The gaa_release_access control_list function releases the access control list bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_list - Access control list to be released, e.g. file, directory, etc.

5.14 gaa_release_access control routine

Purpose:
The gaa_release_access control function releases the access right bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control - Access right to be released, e.g. file, directory, etc.

5.15 gaa_release_access control_set routine

Purpose:
The gaa_release_access control_set function releases the access control set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_set - Access control set to be released, e.g. file, directory, etc.

5.16 gaa_release_access control_list routine

Purpose:
The gaa_release_access control_list function releases the access control list bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_list - Access control list to be released, e.g. file, directory, etc.

5.17 gaa_release_access control_set routine

Purpose:
The gaa_release_access control_set function releases the access control set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_set - Access control set to be released, e.g. file, directory, etc.

5.18 gaa_release_access control routine

Purpose:
The gaa_release_access control function releases the access right bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control - Access right to be released, e.g. file, directory, etc.

5.19 gaa_release_access control_list routine

Purpose:
The gaa_release_access control_list function releases the access control list bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_list - Access control list to be released, e.g. file, directory, etc.

5.20 gaa_release_access control_set routine

Purpose:
The gaa_release_access control_set function releases the access control set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_set - Access control set to be released, e.g. file, directory, etc.

5.21 gaa_release_access control routine

Purpose:
The gaa_release_access control function releases the access right bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control - Access right to be released, e.g. file, directory, etc.

5.22 gaa_release_access control_list routine

Purpose:
The gaa_release_access control_list function releases the access control list bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_list - Access control list to be released, e.g. file, directory, etc.

5.23 gaa_release_access control_set routine

Purpose:
The gaa_release_access control_set function releases the access control set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_set - Access control set to be released, e.g. file, directory, etc.

5.24 gaa_release_access control routine

Purpose:
The gaa_release_access control function releases the access right bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control - Access right to be released, e.g. file, directory, etc.

5.25 gaa_release_access control_list routine

Purpose:
The gaa_release_access control_list function releases the access control list bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_list - Access control list to be released, e.g. file, directory, etc.

5.26 gaa_release_access control_set routine

Purpose:
The gaa_release_access control_set function releases the access control set bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control_set - Access control set to be released, e.g. file, directory, etc.

5.27 gaa_release_access control routine

Purpose:
The gaa_release_access control function releases the access right bound to the ACL.

Parameters:
- acl_handle - A handle to the ACL, returned by the gaa_get_object_acl routine
- access_control - Access right to be released, e.g. file, directory, etc.
Contains various information for further evaluation by the application.

Function value:
GAA status code:
GAA_FAILURE
GAA_NO_CONTEXT
GAA_AUTHORIZED
GAA_NOT_AUTHORIZED
GAA_ADDITIONAL_CHECKS_REQIRED

5.3 gaa_allocate_sec_context routine

Purpose:
Allocate a security context data structure and assign default values.

Parameters:

- minor_status
  Mechanism specific status code
- acl_handle
  A handle bound to a pointer to the security context structure

Function value:
GAA_API status code:
GAA_SUCCESS
GAA_FAILURE

5.4 gaa_release_sec_context routine

Purpose:
Delete a security context. The gaa_delete_sec_context routine will delete the local data structures associated with the specified security context.

Parameters:

- minor_status
  Mechanism specific status code
- sec_context_handle
  A handle bound to a pointer to the security context structure

Function value:
GAA status code:
GAA_SUCCESS
GAA_FAILURE
GAA_NO_BUFFER

5.5 gaa_allocate_buffer routine

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

Parameters:

- minor_status
  Mechanism-specific status code
- buffer
  Pointer to allocated memory for gaa_buffer structure

Function value:
GAA status code:
GAA_SUCCESS
GAA_FAILURE

5.6 gaa_release_buffer routine

Purpose:
Free storage associated with a buffer format name. The storage must have been allocated by a GAA API function. In addition to freeing the storage, the routine will zero the length field in the buffer parameter.

Parameters:

- minor_status
  Mechanism-specific status code

Ryutov/Neuman  Expires February 1999.
The storage associated with the buffer will be deleted.
The gaa_buffer object will not be freed, but its length field will
be zeroed.

Function value:
GAA status code:
GAA_SUCCESS
Successful completion:
GAA_FAILURE
Failure, see minor_status for more information
GAA_NO_BUFFER
No valid buffer was supplied

```c
5.7 gaa_release_tag_value_list routine
Purpose:
Allocate a gaa_tag_value_list data structure and assign default values
Parameters:
minor_status
Mechanism-specific status code
buffer
Pointer to allocated memory for gaa_tag_value_list structure
Function value:
GAA status code:
GAA_SUCCESS
Successful completion:
GAA_FAILURE
Failure, see minor_status for more information
GAA_NO_BUFFER
No valid buffer was supplied
```

```c
5.8 gaa_release_tag_value_list routine
Purpose:
Free storage associated with a buffer
Parameters:
minor_status
Mechanism-specific status code
buffer
The storage associated with the buffer will be deleted
Function value:
GAA status code:
GAA_SUCCESS
Successful completion:
GAA_FAILURE
Failure, see minor_status for more information
GAA_NO_BUFFER
No valid buffer was supplied
```

```c
5.9 gaa_release_tag_vector_list routine
Purpose:
Allocate a gaa_tag_vector_list data structure and assign default values
Parameters:
minor_status
Mechanism-specific status code
buffer
Pointer to allocated memory for gaa_tag_vector_list structure
Function value:
GAA status code:
GAA_SUCCESS
Successful completion:
GAA_FAILURE
Failure, see minor_status for more information
GAA_NO_BUFFER
No valid buffer was supplied
```

```c
5.10 gaa_release_tag_vector_list routine
Purpose:
Free storage associated with a buffer
Parameters:
minor_status
Mechanism-specific status code
buffer
The storage associated with the buffer will be deleted
Function value:
GAA status code:
GAA_SUCCESS
Successful completion:
GAA_FAILURE
Failure, see minor_status for more information
GAA_NO_BUFFER
No valid buffer was supplied
```
5.11 gaa_parse_acl() routine

Purpose:
Parse the ASCII file in the pre-defined format described in
draft-ietf-cat-acc-cntrl-frmw-00.txt into the GAA API internalACL
structures.

Parameters:
minor_status
Mechanism-specific status code

acl_file_handle
A pointer to an ASCII file containing an ACL for the object in
the pre-defined format

acl_handle
A handle to the gaa_acl structure

application_access_rights
A pointer to the list of application-specific names for access
rights. GAA API uses it to build bit-vectors for internal access
rights representation.

This parameter is optional. It is not required if the ACL contains
bit-vector representation of the access rights, for example:
USER  kerberos.v5 joe@ISI.EDU
< FILE: 1100000000000000000000000000000000 >
time_window : 7AM-7PM
time_day : Mon-Fri
< ACCOUNT : 1100000000000000000000000000000000 >
location : *.isi.edu ;

In this case the value is set to GAA_NO_TAG_VALUE_LIST.

Returns:
GAA status code:
GAA_SUCCESS
Successful completion

GAA_FAILURE
Failure, see minor_status for more information

6. The GAA API constants

The following constants are used in GAA API calls and structures,
this list is not complete:

#define GAA_NO_BUFFER             ((gaa_buffer_ptr) 0)
#define GAA_EMPTY_BUFFER          {0, NULL}
#define GAA_NO_DATA               ((gaa_data_ptr)0)
#define GAA_NO_TAG_VALUE_LIST     ((gaa_tag_value_list_ptr)0)
#define GAA_ACL_UNBOUND           ((gaa_acl_ptr)0)
#define NO_EACL_ENTRY             ((gaa_acl_entry_ptr) 0)
#define NO_PRINCIPALS             ((gaa_acl_principal_list_ptr) 0)
#define NO_ACCESS_RIGHTS          ((gaa_acl_access_rights_set_ptr) 0)
#define GAA_NO_CONDITIONS         ((gaa_conditions_ptr)0)

7. The GAA API flags

Flags are 32 bits.

Condition flags:
#define COND_FLG_EVALUATED        0x80000000
#define COND_FLG_SATISFIED        0x40000000
#define COND_FLG_ENFORCE          0x20000000

Flags from 0x10000000 to 0x00000001 are reserved.

8. Acknowledgments

Gene Tsudik, Brian Tung, Bapa Rao, Ilia Ovsiannikov and the Xerox IS
team have contributed to discussion of ideas and material in this
draft.

9. References

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


10. 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

Expires February 1999