Internet-Draft

Tom Parker , ICL
Denis Pinkas , Bull

IETF Common Authentication Technology WG
<draft-ietf-cat-xgssapi-acc-cntrl-03.txt> November 09, 1998

Extended Generic Security Service APIs: XGSS-APIs
Access control and delegation extensions

1. STATUS OF THIS MEMO

This document is an Internet Draft. 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. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress." Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

Comments on this document should be sent to "cat-ietf@MIT.EDU", the IETF Common Authentication Technology WG discussion list.

2. ABSTRACT

The Generic Security Service Application Program Interface (GSS-API), as defined in RFC-1508, provides security services to callers in a generic fashion, supportable with a range of underlying mechanisms and technologies and hence allowing source-level portability of applications to different environments. It defines GSS-API services and primitives at a level independent of underlying mechanism and programming language environment.

The GSSAPI allows a caller application to authenticate a principal identity associated with a peer application, to delegate rights to a peer, and to apply security services such as confidentiality and integrity on a per-message basis.

The primitives of the GSS-API do not currently allow support of security attributes other than a single identity and do not allow fine control of delegation.

The additional primitives described in this document provide support for:

* the exchange of a variety of security attributes, and the

construction of authorization functions using these attributes,
including delegated ones, (attribute handling support functions),

* fine control over delegation by allowing specification of the delegation method, the acceptor(s) of a security context, their type and the restrictions that may apply (acceptor control and support functions).

3. SECURITY ATTRIBUTES

A security attribute is defined as:

```
 SecAttribute ::= {
       attributeType     OBJECT IDENTIFIER,
       definingAuthority OCTET STRING        OPTIONAL,
       securityValue     OCTET STRING }
```

attributeType

Defines the type of the attribute. Attributes of the same type have the same authorization semantics.

definingAuthority

It indicates the authority responsible for defining the value within the attribute type. Some policies demand that multiple sources of values for a given attribute type be supported (e.g. a policy accepting attribute values defined outside the security domain), These policies give rise to a risk of value clashes. The definingAuthority field is used to separate these values. When not present, the value defaults to the name of the authority that issued the attribute.

securityValue

The value of the security attribute. Its syntax is determined by the attribute type.

Security attributes are composed of principal attributes and of qualifier attributes.

3.1. PRINCIPAL ATTRIBUTES

Principal attributes are composed of security privileges and miscellaneous attributes.

3.1.1. PRIVILEGES ATTRIBUTES

Security privileges are defined as security attributes attached to a principal, and only usable for access control purposes. Ones defined here are: access identity, group memberships, clearance and capability. The use of OBJECT IDENTIFIERS allows for other types to be standardised.

3.1.2. MISCELLANEOUS ATTRIBUTES

Miscellaneous attributes are defined as security attributes attached to a principal, which are not security privileges. They are used for a variety of purposes. Ones defined here are: the domain name of the issuer of the security attributes, an audit identity, restrictions and validity time periods.

3.2. QUALIFIER ATTRIBUTES

Qualifier attributes describe the context acceptors for which controls are to apply. Ones defined here are: acceptor name and application trust group.

3.3. ATTRIBUTES DEFINITIONS

3.3.1. PRIVILEGE ATTRIBUTES

A privilege attribute is attached to a principal and is only usable for access control purposes. Privileges are defined under the OID:

```
privilege-attribute    OBJECT IDENTIFIER ::=
{ iso(1) identified-organisation(3) icd-ecma(012) technical-report(1)
security-in-open-systems(046) privilege-attribute(4) }
```

3.3.1.1. ROLE ATTRIBUTE

The role attribute represents the principal's role. The type of this attribute is:

```
{ privilege-attribute 1 }
```

The type and the value of this attribute can be set and returned.

3.4.1.2. ACCESS IDENTITY

The access identity represents an identity to be used for access control purposes. A security context or a credential may not contain more than a single access identity for a given principal. This attribute does not need to be present. The type of this attribute is :

```
{ privilege-attribute 2 }
```

The type and the value of this attribute can be set and returned.

3.4.1.3. PRIMARY GROUP

The primary group represents a uniquely prominent group to which a principal belongs. A security context or a credential may not contain more than one primary group for a given principal. The type

of this attribute is :

{ privilege-attribute 3 }

The type and the value of this attribute can be set and returned.

3.4.1.4. GROUP

A group represents a characteristic common to several principals. A security context or a credential may contain more than one group for a given principal. The type of this attribute is :

{ privilege-attribute 4 }

The type and the value of this attribute can be set and returned.

3.4.1.5. CAPABILITY

A capability nominates a resource and the operation(s) that can be performed on that resource. The type of this attribute is :

{ privilege-attribute 5 }

The type and the value of this attribute can be set and returned.

3.3.2. MISCELLANEOUS ATTRIBUTES

Miscellaneous attributes are defined under the OID:

misc-attributeOBJECT IDENTIFIER ::=
{ iso(1) identified-organisation(3) icd-ecma(012) technical-report(1) security-in-open-systems(046) misc-attribute(3) }

3.4.2.1. AUDIT IDENTITY

The access identity represents the principal's identity to be used for audit purposes. The type of this attribute is :

{ misc-attribute 2 }

Only the type of this attribute can be set and both the type and the value can be returned.

3.4.2.2. ISSUER DOMAIN NAME

The issuer domain name represents the name of the domain that has issued the security attributes. It cannot be set by a call to GSS_Set_cred_attributes. The type of this attribute is :

{ misc-attribute 10 }

Only the type of this attribute can be set and both the type and the value can be returned. This attribute may always be returned by a call to GSS_Get_cred_attributes.

3.4.2.3. VALIDITY PERIODS

The validity periods represent a list of the time periods within which the security attributes are valid. The type of this attribute is :

{ misc-attribute 11 }

The type and the values of this attribute can be set and returned.

3.4.2.4. OPTIONAL RESTRICTIONS

The Optional restrictions represent restrictions that apply to the security context. The context may be accepted, even if the application is unable to understand the optional restrictions. The type of this attribute is :

{ misc-attribute 12 }

The type and the values of this attribute can be set and returned.

3.4.2.5. MANDATORY RESTRICTIONS

The mandatory restrictions represent restrictions that apply to the security context. The context must not be accepted if the application is unable to understand the mandatory restrictions. The type of this attribute is :

{ misc-attribute 13 }

The type and the values of this attribute can be set and returned.

3.3.3. QUALIFIER ATTRIBUTES

Qualifier attributes are security attributes which define which applications are authorized to be a security context acceptor. In addition to the qualifier attribute it is possible to specify whether delegation is authorized or not for the context acceptor.

Qualifier attributes describe the context acceptors for which controls are to apply. Qualifier attributes are defined under the OID:

```
qualifier-attribute    OBJECT IDENTIFIER ::=
{ iso(1) identified-organisation(3) icd-ecma(012) technical-report(1)
security-in-open-systems(046) qualifier-attribute(5) }
```

3.4.3.1. ACCEPTOR NAME

An acceptor name represents the name of an application that can

potentially accept the security context. The type of this attribute is :

{ qualifier-attribute 1 }

3.4.3.2. APPLICATION TRUST GROUP

An application trust group represents a group of acceptors, defined by the security administrator, that mutually trust each other not to spoof each others' identity. The type of this attribute is :

{ qualifier-attribute 2 }

4. ATTRIBUTE SET REFERENCE

Attribute set references are defined under the OID:

attribute-set-reference OBJECT IDENTIFIER ::=
{ iso(1) identified-organisation(3) icd-ecma(012) technical-report(1) security-in-open-systems(046) attribute-set-reference (9)

An Attribute set reference is used to select a set of security attributes and acceptor controls according to policy. At present only a role name is defined.

4.1. ROLE NAME

The role name is an attribute set reference used to select a set of security attributes. The type of this attribute is :

{ attribute-set-reference 1 }

The type and the values of this reference can only be set.

6. INTERFACE DESCRIPTIONS

The interfaces are split between attribute handling support functions and context acceptor control and support functions.

6.1. ATTRIBUTE HANDLING SUPPORT FUNCTIONS

Three attribute handling support functions are defined :

6.1.1. GSS_Set_cred_attributes

To enable a GSS-API context initiator to specify security attributes, i.e. security privileges and miscellaneous security attributes to be included in the caller credentials in order to become part of a security context.

6.1.2. GSS_Get_sec_attributes

To extract security attributes, either from a GSS-API context, or from a credential handle. Both privilege attributes and

miscellaneous attributes can be retrieved. The function can be invoked either by a context initiator or by a context acceptor. When

applied to a GSS-API context and invoked by a context acceptor, if delegation is being used, the privilege and miscellaneous attributes which are returned and only those of the initiator, i.e. the initiator of the delegation chain.

6.2. CONTEXT ACCEPTOR SUPPORT FUNCTION

There is a single function.

6.2.1. GSS_Get_received_creds

To extract credential handles from a GSS-API context (established with GSS_Accept_sec_context function). This is only applicable for some forms of delegation supporting multiple incoming credentials, and can only be invoked by a context acceptor. A call to GSS_Get_received_creds is an intermediary step for an acceptor, before extracting the security attributes of each set of credentials with a call to GSS_Get_sec_attributes. The credential handles are ordered. They start with the credentials of the first initiator and finish with the the credentials of the immediate invoker.

6.3. CONTEXT ACCEPTOR CONTROL FUNCTIONS

These functions enable a GSS-API context initiator to impose constraints on the security context to be established via GSS_Init_sec_context function, and enable a GSS-API context acceptor to retrieve the control information that applies to a security context established using the GSS_Accept_sec_context function, and to build credentials from others.

Three context acceptor control functions are defined:

6.3.1. GSS_Set_cred_controls function

To enable a GSS-API context initiator to specify acceptor controls to be included in the caller's credentials in order to be part of a security context. The controls determine the context acceptors with which valid security contexts can be established using the associated credentials, and whether they can act only as targets only, delegates only as or as delegate/targets.

* an acceptor designated as being a target only may use the privileges attributes in the received credentials when authorising access to its own protected resources and may not forward them.

* an acceptor designated as being a delegate only may use the privilege attributes in the received credentials to forward them but should not use them when authorising access to its own protected resources.

Note: Only the acceptor system's AEF (Access Enforcement Facility as described in ISO/IEC 10181-3: The Access Control Framework) can prevent an acceptor permitting access based on

attributes not intended for it. However it is not in the interests of an acceptor or its AEF to permit access to resources under their control on the basis of attributes that are explicitly stated as not being appropriate.

* an acceptor designated as being both a target and a delegate may use the privilege attributes in the received credentials when authorising access to its own protected resources and may also forward them.

Restrictions over the operations that are authorised under the context can also be specified.

6.3.2. GSS_Get_sec_controls function

To enable a GSS-API context initiator or a GSS-API context acceptor to extract acceptor control information either from a credential handle or from a security context.

6.3.3. GSS_Compound_creds function

To enable a delegate (which is acting as a GSS-API target for a context initiator, and as a GSS-API context initiator for another delegate or target) to build new credentials made from received credentials and its own credentials.

7. DETAILED DESCRIPTION OF THE CALLS

7.1. ATTRIBUTE HANDLING CALLS

7.1.1. GSS_Set_cred_attributes call

Input :

- cred_handle OCTET STRING,
- required_attributes SET OF SecAttribute,
- new_cred_req BOOLEAN
- commit_cred_req BOOLEAN

Output :

- output_cred_handle OCTET STRING

Return major_status code:

- GSS_S_COMPLETE indicates that the nominated attributes are permitted to the caller and have been set.
- GSS_S_CREDENTIALS_EXPIRED indicates that the specified credentials have expired.

- GSS_S_DEFECTIVE_CREDENTIAL indicates that defective
  credentials have been detected.
- GSS_S_FAILURE indicates a failure,unspecified
  at the GSS-API level.

- GSS_S_UNAUTHORIZED indicates that the function, or an argument of the function was not authorised.
- GSS_S_UNAVAILABLE indicates that the operation is not supported.

This function enables a caller to request a set of privileges and miscellaneous attributes, optionally replacing existing credentials or creating a new set. The effect of this interface is not cumulative, the requested attributes replace any existing attributes in the credentials claimed.

Parameters for GSS_Set_cred_attributes:

cred_handle

Handle for credentials claimed, cred_handle refers to an authenticated principal. Supply NULL to use default credentials.

required_attributes

A set of required privilege and miscellaneous attributes. NULL specifies default attributes to be requested. Otherwise, only the privilege and miscellaneous attributes specified will be present.

If a specified attribute is provided with a NULL value field, the value allocated to the attribute will be the default for the specified attribute available to the authenticated principal according to the prevailing security policy. Otherwise the value specified will be that present. If a value specified clashes with policy, an error is returned.

If an attribute set reference (e.g.a role name) is specified as a single attribute required, and policy permits the principal to use it, it will be used as an attribute set reference to select a set of attributes and acceptor controls according to policy.

If an attribute set reference (e.g.a role name) is specified along with other required attributes, and policy permits the principal to use the role name, the attributes potentially available for the authenticated principal are taken from a set compounded of the principal's authorised attributes, and the attributes associated with the role name.

new_cred_req

TRUE for a new credentials set, FALSE replaces the original.

commit_cred_req

TRUE for immediate attribute acquisition, FALSE for deferred attribute acquisition.

output_cred_handle

The credentials handle for the changed or new credentials. GSS_Set_cred_attributes produces a modified version of the input credentials (cred_handle). The original credentials are changed if new_cred_req is FALSE, otherwise the output_cred_handle references a new, and different, copy of the original input credentials (which remain untouched). GSS_Release_cred can be used when the caller is finished with any new credentials created by this function.

7.1.2. GSS_Get_sec_attributes call

Input :

- cred_handle OCTET STRING,
- context_handle INTEGER,
- attribute_types_required SET OF OBJECT IDENTIFIER

Output :

- priv_attributes SET OF SecAttribute
- misc_attributes SET OF SecAttribute
- other_cred_present BOOLEAN

Return major_status code :

- GSS_S_COMPLETE	indicates that retrieval of attributes is supported and that all, some, or none of the requested attribute types have been returned.
- GSS_S_CONTEXT_EXPIRED	indicates that the specified security context has expired.
- GSS_S_CREDENTIALS_EXPIRED	indicates that the specified credentials have expired.
- GSS_S_DEFECTIVE_CREDENTIAL	indicates that defective credentials have been detected.
- GSS_S_FAILURE	indicates a failure, unspecified at the GSS-API level.
- GSS_S_UNAVAILABLE	indicates that the operation is not supported.

This function can be used by context initiators and context acceptors to query attributes in credentials or security contexts. If the credentials or security context represents a delegation chain, attributes are retrieved only from the initiator of the chain. If the attribute_types_required parameter is not supplied,

then all attribute types are returned. This option could allow clients of this interface to query all attributes and pass privilege attributes to a separate authorization service to make a decision.

To obtain security attributes from intermediates in a delegation chain, the caller should first call GSS_Get_received_creds (see section 5.2.1 and section 6.2.1).

Parameters for GSS_Get_sec_attributes:

cred_handle

Handle to credentials, cred_handle refers to an authenticated principal. Supply NULL to use default credentials, or a context handle. Note that NULL, without a context handle, is only used for obtaining the caller's own attributes.

context_handle

GSS-API security context handle, context_handle refers to an established security context. Context_handle is ignored if a non-NULL cred_handle is presented. (Note: it is typically only necessary to use a context_handle parameter rather than cred_handle for the case when a security context is emitted by gss_accept_sec_context, but not with an accompanying set of delegated credentials).

attribute_types_required

A set of security attribute types. If the default (NULL) is specified, then all miscellaneous and privilege attribute types are returned.

This standard does not specify which attributes must be supported, but some common security attributes are defined in section 2.

priv_attributes

A set of privilege attributes. Response is conditional on the "attribute_types_required" input.

misc_attributes

A set of miscellaneous attributes. Response is conditional on the "attribute_types_required" input.

other_cred_present

TRUE when the caller is a context acceptor querying a security context and when more than one set of credentials is present. If interested in the other credential(s), the caller should next call GSS_Get_received_creds,

FALSE when either the caller is a context initiator or when the caller is a context acceptor querying a security context and when no

other credential is present.

7.2. CONTEXT ACCEPTOR SUPPORT FUNCTION

7.2.1. GSS_Get_received_creds call

Input :

- context_handle INTEGER,

Output :

- received_creds SEQUENCE OF OCTET STRING

Return major_status code :

- GSS_S_COMPLETE	indicates that the requested delegate credentials were retrieved.
- GSS_S_CONTEXT_EXPIRED	indicates that the specified security context has expired.
- GSS_S_FAILURE	indicates a failure, unspecified at the GSS-API level.
- GSS_S_UNAUTHORIZED	indicates that the function, or an argument of the function was not authorised.
- GSS_S_UNAVAILABLE	indicates that the operation is not supported.

This function supports the retrieval of all credentials received by an acceptor. It is intended for context acceptors that require not only the initiator's credentials, but also delegates' credentials, to apply their local security policy. A typical example is the retrieval of delegate credentials to subsequently obtain delegate privilege attributes (using GSS_Get_sec_attributes) for use in authorization decisions.

Parameters for GSS_Get_received_creds:

context_handle

GSS-API security context handle, context_handle refers to a security context that is part of an established association. A default context is assumed if no context_handle is supplied.

received_creds

Contains an ordered list of credentials for the original initiator and for each of the intermediate delegates (if any) between the original initiator and this context acceptor, the first of these being the credentials of the original initiator, and the last being

of the immediately preceding delegate. It is expected that the normal use for such credentials would merely be inspection via GSS_Get_sec_attributes as most known mechanisms would not permit

such delegate credentials to be directly used for initiating further security contexts. Note that it is the caller's responsibility to free any received credentials returned from gss_get_received_creds via gss_release_cred.

7.3. ACCEPTOR CONTROL HANDLING CALLS

The following construct is used in both the GSS_Set_cred_controls and GSS_Get_sec_controls calls:

```
AcceptorControl ::=  SEQUENCE {
     targetOnly        SEQUENCE OF SecAttribute OPTIONAL,
     delegateOnly      SEQUENCE OF SecAttribute OPTIONAL,
     delegateTarget    SEQUENCE OF SecAttribute OPTIONAL,
     delegationMode    DelegationMode OPTIONAL,}

DelegationMode ::=  ENUMERATED {
     default           (0),
     simple            (1),
     composite         (2),
     traced            (3),}
```

The fields targetOnly, delegateOnly and delegateTarget specify one or several qualifier attributes describing the acceptors (as targets, delegates or delegate/targets) for which controls are to apply.

* the targetOnly specifies that the qualifier(s) are identifying one or more targets, none of which may use the credentials as a delegate.

* the delegateOnly choice specifies that the qualifier(s) are identifying one or more delegates, none of which should use the privilege attributes in the credentials when authorising access to their own protected resources, but which may use the received credentials as a delegate.

* the delegateTarget choice specifies that the qualifier is identifying one or more delegate/targets any of which can use the received credentials as a delegate and can also use the privileges attributes in the the credentials when authorising access to its own protected resources.

delegationMode

Indicates the mode of delegation required.

Currently three delegation modes and one default are specified:

- default: whatever mode of delegation has been set as default (this

may be no delegation).

- simple: only the original initiator's credentials have to be

forwarded in the security context being established,

- composite: the credentials of the original initiator and of the immediate caller have to be forwarded,

- traced: the credentials of the original initiator, of all the delegates, including the immediate caller have to be forwarded.

7.3.1. GSS_Set_cred_controls call

Input :

- cred_handle	OCTET STRING,
- required_acceptor_control	AcceptorControl,
- replace_old_controls	BOOLEAN
- new_cred_req	BOOLEAN
- commit_cred_req	BOOLEAN

Output :

- output_cred_handle	OCTET STRING

Return major_status code:

- GSS_S_COMPLETE	indicates that the controls have been set.
- GSS_S_CREDENTIALS_EXPIRED	indicates that the specified credentials have expired.
- GSS_S_DEFECTIVE_CREDENTIAL	indicates that defective credentials have been detected.
- GSS_S_FAILURE	indicates a failure, unspecified at the GSS-API level.
- GSS_S_UNAUTHORIZED	indicates that the function, or an argument of the function was not authorised.
- GSS_S_UNAVAILABLE	indicates that the operation is not supported.

This function supports requests to set context acceptor controls, optionally replacing existing credentials controls or creating a new set of credentials with new controls. The effect of this interface is either cumulative or not depending on the value of the replace_old_controls parameter.

Parameters for GSS_Set_cred_controls:

cred_handle

Handle for credentials claimed, it refers to an authenticated
principal. Supply NULL to use default credentials.

required_acceptor_control

The control settings required.

replace_old_controls

TRUE to replace acceptor controls existing in original credentials. FALSE to specify additional controls.

new_cred_req

TRUE for a new credentials set, FALSE to modify the original.

commit_cred_req

TRUE for immediate action, FALSE for deferred action.

output_cred_handle

GSS_Set_cred_controls produces a modified version of the input credentials (cred_handle). The original credentials are directly changed if duplicate_cred_req is FALSE, otherwise the output_cred_handle references a new, and potentially different, copy of the original input credentials (which remain untouched). gss_release_cred can be used when the caller is finished with any new credentials created by this function.

7.3.2. GSS_Get_sec_controls call

Input :

- cred_handle OCTET STRING,
- context_handle INTEGER,

Output :

- acceptor_controls SET OF AcceptorControl,

Return major_status code :

- GSS_S_COMPLETE	indicates that the acceptor control information has been returned
- GSS_S_CREDENTIALS_EXPIRED	indicates that the specified credentials have expired.
- GSS_S_DEFECTIVE_CREDENTIAL	indicates that defective credentials have been detected.
- GSS_S_FAILURE	indicates a failure, unspecified at the GSS-API level.

- GSS_S_UNAVAILABLE indicates that the operation is not supported.

This function enables a caller to enquire the current value of the acceptor controls in the specified credentials or context. This function can be used by context initiators and context acceptors to query acceptor controls in credentials or security contexts.

Parameters for GSS_Get_sec_controls:

cred_handle

Handle to credentials. It refers to an authenticated principal. Supply NULL to use default credentials, or a context handle.

context_handle

GSS-API security context handle, context_handle refers to a security context that is part of an established association. Context_handle is ignored if a non-NULL cred_handle is presented. (Note: it is typically only necessary to use a context_handle parameter rather than cred_handle for the case when a security context is emitted by gss_accept_sec_context, but not with an accompanying set of delegated credentials).

acceptor_controls

A set of acceptor controls. Acceptor controls are described in section 6.2.

7.3.3. GSS_Compound_creds call

Input :

- delegated_cred_handle OCTET STRING
- cred_handle OCTET STRING,

Output :

- cred_handle_new OCTET STRING

Return major_status code :

GSS_S_COMPLETE	indicates that the credentials were successfully compounded
- GSS_S_CREDENTIALS_EXPIRED	indicates that one or more of the specified credentials have expired.
- GSS_S_DEFECTIVE_CREDENTIAL	indicates that defective credentials have been detected.
- GSS_S_FAILURE	indicates a failure, unspecified at the GSS-API

level.

- GSS_S_UNAVAILABLE indicates that the operation is not supported.

Parameters for gss_compound_cred:

delegated_cred_handle

A handle to the credentials being delegated, it refers to one or several authenticated principals.

cred_handle

A handle to claimed credentials of the caller, cred_handle refers to an authenticated principal.

cred_handle_new

A handle to the compounded set of credentials.

8. C-LANGUAGE BINDINGS

This section specifies C language bindings for the extended GSS-API functions.

8.1. DATA TYPES AND CALLING CONVENTIONS

The following data types :

- OM_uint32,
- gss_buffer_t,
- gss_OID,
- gss_OID_set,
- gss_cred_id_t,
- gss_ctx_id_t,

are defined in [RFC-1508], along with the calling conventions.

8.1.1. SECURITY ATTRIBUTES

A security attribute (see section 2) has the following data structure:

```
typedef struct gss_sec_attr_desc  {
       gss_OID              attribute_type;
       gss_buffer_t         defining_authority;
                            /* specify GSS_C_NO_BUFFER when
                            non present */
       gss_buffer_t         security_value;
       } gss_sec_attr;
```

8.1.2. SECURITY ATTRIBUTE SETS

A set of security attributes has the following structure:

```
typedef struct gss_sec_attr_set_desc  {
       OM_uint32              attribute_count;
       gss_sec_attr*          attributes;
       } gss_sec_attr_set;
```

The attribute_count field contains the number of security attributes in the set.

8.1.3. CREDENTIALS LIST

A list of credentials has the following structure:

```
typedef struct  {
       OM_uint3               cred_count;
       gss_cred_id_t*         cred_list;
       } gss_cred_list;
```

The cred_count field contains the number of credentials in the list.

8.1.4. ACCEPTOR CONTROL

Acceptor control has the following structure:

```
typedef struct gss_acceptor_control_desc  {
       gss_sec_attr      target_only;
                         /* specify GSS_C_NULL_SEC_ATTR when
                         non present */
       gss_sec_attr      delegate_only;
                         /* specify GSS_C_NULL_SEC_ATTR when
                         non present */
       gss_sec_attr      delegate_target;
                         /* specify GSS_C_NULL_SEC_ATTR when
                         non present */
       OM_uint32         delegation_mode;
                         /* specify NULL when non present */
       } gss_acceptor_control;
```

8.1.5. ACCEPTOR CONTROL SET

A set of Acceptor Control has the following structure :

```
typedef struct gss_control_set_desc  {
       OM_uint32              control_count;
       gss_acceptor_control*  acceptor_controls;
       } gss_control_set;
```

The control_count field contains the number of acceptor controls in the set.

8.1.6. IDENTIFIER

Identifiers have the following data structure:

```
typedef struct  {
       gss_type_en             id_type
       gss_value               id_value;
       } gss_id;
```

Where id_type identifies the syntax within the Identifier type:

```
typedef enum  {
       gss_oid_t,              /* for OID */
       gss_integer,            /* for Integer */
       gss_string,             /* for character string */
       gss_uuid,               /* for DCE UUID */
       gss_buffer_t;           /* for gss_buffer */
       } gss_type_en;
```

And where id_value is the actual value of the data of type Identifier:

```
struct union  {
       gss_OID                 OID;
       OM_uint32*              integer;
       char*                   string;
       uuid_t*                 uuid;
       gss_buffer_t            buffer;
       } gss_value;
```

This C type is applicable for the following types of attribute: access identity, primary group, capability, audit identity, issuer domain name, and role name.

When one of these attributes is handled in a call, the security_value field of the gss_sec_attr structure for this attribute contains a pointer to the gss_id structure.

8.1.7. IDENTIFIER SET

Identifier sets have the following data structure:

```
typedef struct gss_id_set_desc  {
       OM_uint32               id_count;
       gss_id*                 ids;
       } gss_id_set;
```

The id_count field contains the number of Identifiers in the set.

This C type is applicable for the following types of attribute: group, role, optional restrictions, mandatory restrictions, acceptor name and application trust group.

When one of these attributes is handled in a call, the security_value field of the gss_sec_attr structure for this

attribute contains a pointer to the gss_id_set structure.

8.1.8. TIME PERIOD

A time period has the following structure:

```
typedef struct gss_time_period_desc  {
     time_t           start_time;
                      /* NULL for unconstrained start time */
     time_t           end_time;
                      /* NULL for unconstrained end time */
     } gss_time_period;
```

8.1.9. TIME PERIODS LIST

Time period lists have the following data structure:

```
typedef struct gss_period_list _desc  {
     OM_uint32                period_count;
     gss_time_period*         periods;
     } gss_period_list;
```

The period_count field contains the number of time periods in the list.

This C type is applicable for the miscellaneous attribute: time period.

When a list of time periods is returned by a GSS_Get_sec_attributes call, or set by a GSS_Set_cred_attributes call, the security_value field of the gss_sec_attr structure in gss_sec_attr_set contains a pointer to the gss_period_list_structure.

8.2. XGSS-API ROUTINE DESCRIPTIONS

8.2.1. gss_set_cred_attributes

```
/* set attributes values in credentials */
OM_uint32 gss_set_cred _attributes (
       gss_cred_id_t           cred_handle,               /* IN */
       gss_sec_attr_set        required_attributes,       /* IN */
       OM_uint32               new_cred_req,              /* IN */
       OM_uint32               commit_cred_req,           /* IN */
       OM_uint32*              minor_status,              /* OUT*/
       gss_cred_id_t*          output_cred_handle);       /* OUT*/
```

8.2.2. gss_get_sec_attributes

```
/* get attributes associated with credentials or security context */
OM_uint32 gss_get_sec_attributes (
       gss_cred_id_t           cred_handle,               /* IN */
       gss_ctx_id_t            context_handle,            /* IN */
```

```
gss_OID_set             attribute_types_required,   /* IN */
OM_uint32*              minor_status,               /* OUT*/
gss_sec_attr_set**      priv_attributes,            /* OUT*/
```

```
        gss_sec_attr_set**      misc_attributes);          /* OUT*/
        OM_uint32               other_cred_present         /* OUT*/
```

8.2.3. gss_get_received_creds

```
/* get received credentials associated with a security context */
OM_uint32 gss_get_received_creds (
       gss_ctx_id_t            context_handle,            /* IN */
       OM_uint32*              minor_status,              /* OUT*/
       gss_cred_list**         received_creds);           /* OUT*/
```

8.2.4. gss_set_cred_controls

```
/* Set acceptor controls in credentials for context establishment
*/
OM_uint32 gss_set_cred_controls (
       gss_cred_id_t           cred_handle,               /* IN */
       gss_ control_set        required_control,          /* IN */
       OM_uint32               replace_old_controls,      /* IN */
       OM_uint32               new_cred_req,              /* IN */
       OM_uint32               commit_cred_req,           /* IN */
       OM_uint32*              minor_status,              /* OUT*/
       gss_cred_id_t*          output_cred_handle);       /* OUT*/
```

8.2.5. gss_get_sec_controls

```
/* set context acceptor controls on credentials */
OM_uint32 gss_get_sec_controls (
       gss_cred_id_t           cred_handle,               /* IN */
       gss_ctx_id_t            context_handle,            /* IN */
       OM_uint32*              minor_status,              /* OUT*/
       gss_control_set*        acceptor_controls);        /* OUT*/
```

8.2.6. gss_compound_cred

```
/* compound credentials for delegation */
OM_uint32 gss_compound_cred (
       gss_cred_id_t           delegated_cred_handle,     /* IN */
       gss_cred_id_t           cred_handle,               /* IN */
       OM_uint32*              minor_status,              /* OUT*/
       gss_cred_id_t           cred_handle_new);          /* OUT*/
```

9. ACKNOWLEDGEMENTS

Acknowledgements are due to the following people : Eric Baize, Belinda Fairthorne, Stephen Farell, Jacques Lebastard and Tom Parker for providing material for the construction of this document and/or providing useful inputs.

10. SECURITY CONSIDERATIONS

Security issues are discussed throughout this memo.

11. REFERENCES

[RFC 1508] Generic Security Service API, J Linn,
September 1993

[RFC 1509] Generic Security Service API : C-bindings, J Wray,
September 1993

12. AUTHORS'S ADDRESSES

Tom Parker Internet email: t.a.parker@win0199.wins.icl.co.uk
ICL Enterprises Telephone: +44.1344.472169
59 Old Road, Fax : +44.1249.822703
Derry Hill,
Calne,
Wiltshire SN11 9NF,
United Kingdom

Denis Pinkas Internet email: Denis.Pinkas@bull.net
Bull Telephone: +33 1 30 80 34 87
Rue Jean-Jaures Fax: +33 1 30 80 33 21
BP 68
78340 Les Clayes-sous-Bois
FRANCE

13. CONTENT LIST