System i

Programming
Generic Security Services APIs

Version 6 Release 1

System i

Programming
Generic Security Services APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in "Notices," on page 77.

This edition applies to version 6, release 1, modification 0 of IBM i5/OS (product number 5761-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Generic Security Services APIs

The Generic Security Services (GSS) APIs support job environments for most EBCDIC CCSIDs. CCSID 290 and 5026 are not supported because of the variance of lowercase letters a to z.

The GSS APIs provide security services to applications that use peer-to-peer communications. For more information, see the Network authentication service topic collection.

The GSS APIs are:

- "gss_accept_sec_context()—Accept Security Context" on page 3 (Accept security context) accepts a security context created by the context initiator.
- "gss_acquire_cred()—Acquire GSS Credential" on page 7 (Acquire GSS credential) allows an application to acquire a GSS credential.
- "gss_add_cred()—Add Credential Element to Existing GSS Credential" on page 10 (Add credential element to existing GSS credential) adds a credential element to an existing GSS credential.
- "gss_add_oid_set_member()—Add OID to an OID Set" on page 12 (Add OID to an OID set) adds a new OID to an existing OID set.
- "gss_canonicalize_name()—Reduce GSS Internal Name to Mechanism Name" on page 13 (Reduce GSS internal name to mechanism name) takes a GSS internal name that contains multiple internal representations and returns a new GSS internal name with a single name representation that corresponds to the specified security mechanism.
- "gss_compare_name()—Compare Two Internal GSS Names" on page 15 (Compare two internal GSS names) allows an application to compare two internal names to determine whether they refer to the same object.
- "gss_context_time()—Get Number of Seconds Security Context Remains Valid" on page 16 (Get number of seconds security context remains valid) checks the specified security context and returns the number of seconds that the context remains valid.
- "gss_create_empty_oid_set()—Create Empty OID Set" on page 17 (Create empty OID set) creates a new, empty OID set. Members can be added to the OID set by calling the gss_add_oid_set_member() routine.
- "gss_delete_sec_context()—Delete Security Context" on page 18 (Delete security context) deletes one end of a security context.
- "gss_display_name()—Get Textual Representation of Internal GSS Name" on page 19 (Get textual representation of internal GSS name) returns the textual representation of an opaque internal name.
- "gss_display_status()—Get Textual Representation of GSS Status Code or Mechanism Code" on page 21 (Get textual representation of GSS status code or mechanism code) provides an application with a textual representation of a GSS or mechanism status code.
- "gss_duplicate_name()—Create Duplicate GSS Internal Name" on page 22 (Create duplicate GSS internal name) creates a duplicate of a GSS internal name.
- "gss_export_cred()—Export GSS Credential" on page 23 (Export GSS Credential) creates a credential token for a GSS-API credential.
- "gss_export_name()—Create Opaque Token for a Mechanism Name" on page 25 (Create Opaque Token for a Mechanism Name) creates an opaque token for a mechanism name.
- "gss_export_sec_context()—Export Security Context" on page 26 (Export Security Context) creates a context token for a GSS API security context.
- "gss_get_mic()—Generate Cryptographic Signature for Message" on page 27 (Generate cryptographic signature for message) generates a cryptographic signature for a message and returns this signature in a token that can be sent to a partner application.

- “gss_import_cred()—Import GSS Credential” on page 28 (Import GSS Credential) accepts a credential token created by the**gss_export_cred()**routine and creates a GSS API credential.
- “gss_import_name()—Convert Printable Name to GSS Internal Format” on page 30 (Convert printable name to GSS internal format) converts a printable name to the GSS internal format.
- “gss_import_sec_context()—Import Security Context” on page 31 (Import Security Context) accepts a security context token created by the**gss_export_sec_context()**routine and creates a GSS API security context.
- “gss_indicate_mechs()—Determine Available Security Mechanisms” on page 32 (Determine available security mechanisms) allows an application to determine which security mechanisms are available on the local system.
- “gss_init_sec_context()—Initiate Security Context” on page 33 (Initiate security context) initiates a security context for use by two communicating applications.
- “gss_inquire_context()—Get Information About Security Context” on page 38 (Get information about security context) returns information about a security context to the calling application.
- “gss_inquire_cred()—Get Information About GSS Credential” on page 40 (Get information about GSS credential) returns information about a GSS credential to the calling application.
- “gss_inquire_cred_by_mech()—Get Information About GSS Credential for Single Security Mechanism” on page 41 (Get information about GSS credential for single security mechanism) returns information about a GSS credential for a single security mechanism.
- “gss_inquire_mechs_for_name()—Determine Mechanisms to Process Name” on page 43 (Determine mechanisms to process name) returns the mechanisms with which a name may be processed.
- “gss_inquire_names_for_mech()—Get Name Types Supported by Security Mechanism” on page 44 (Get name types supported by security mechanism) returns the name types supported by a security mechanism.
- “gss_krb5_acquire_cred_cache()—Acquire GSS Credential from a Kerberos Protocol Credentials Cache” on page 47 (Acquire GSS Credential from a Kerberos Protocol Credentials Cache) acquires a GSS API credential using a Kerberos credentials cache.
- “gss_krb5_ccache_name()—Set Default Kerberos Protocol Credentials Cache Name” on page 48 (Set Default Kerberos Protocol Credentials Cache Name) sets the default credentials cache name for use by the Kerberos mechanism.
- “gss_krb5_copy_ccache()—Copy Tickets From Associated GSS Credentials to Kerberos Protocol Credentials Cache” on page 49 (Copy Tickets From Associated GSS Credentials to Kerberos Protocol Credentials Cache) copies the tickets from the Kerberos credentials cache associated with a GSS API credential to a credentials cache provided by the caller.
- “gss_krb5_get_ccache()—Get Kerberos Protocol Credentials Cache Associated with Specified GSS Credential” on page 45 (Get Kerberos protocol credentials cache associated with specified GSS credential) returns the returns the handle for the Kerberos credentials cache associated with a GSS credential.
- “gss_krb5_get_tkt_flags()—Get Kerberos Protocol Ticket Flags” on page 50 (Get Kerberos protocol ticket flags) returns the Kerberos ticket flags from the Kerberos ticket associated with the security context.
- “gss_oid_to_str()—Convert OID Object to String Representation of Object” on page 51 (Convert OID object to string representation of object) converts a gss_oid object to a string representation of the object identifier.
- “gss_process_context_token()—Process Received Context Token” on page 53 (Process received context token) processes a context token received from the partner application.
- “gss_release_buffer()—Release Storage Associated with Buffer” on page 54 (Release storage associated with buffer) releases storage associated with a gss_buffer_t buffer. The gss_buffer_desc structure itself is not released.
- “gss_release_cred()—Release Storage Associated with GSS Credential” on page 55 (Release storage associated with GSS credential) releases the local data structures associated with a GSS credential.

- “gss_release_name()—Release Storage Associated with GSS Internal Name” on page 56 (Release storage associated with GSS internal name) releases storage associated with a gss_name_t internal name.
- “gss_release_oid()—Release Storage Associated with OID Object” on page 57 (Release storage associated with OID object) releases storage associated with a gss_oid object.
- “gss_release_oid_set()—Release Storage Associated with a Set of OID Objects” on page 58 (Release storage associated with a set of OID objects) releases storage associated with a gss_oid_set object.
- “gss_str_to_oid()—Convert String Representation of an Object Identifier to an Internal OID Object” on page 59 (Convert string representation of an object identifier to an internal OID object) converts the string representation of an object identifier to a gss_OID object.
- “gss_test_oid_set_member()—Determine if Specified OID is Contained in a Specified OID Set” on page 60 (Determine if specified OID is contained in a specified OID set) checks an oid set to see if a specified oid is a member of the set.
- “gss_unwrap()—Unwrap a Message” on page 61 (Unwrap a message) unwraps a message sealed by the gss_wrap() routine and verifies the embedded signature.
- “gss_verify_mic()—Verify that Cryptographic Signature is Correct” on page 63 (Verify that cryptographic signature is correct) verifies that the cryptographic signature for a message is correct.
- “gss_wrap()—Cryptographically Sign and Optionally Encrypt Message” on page 65 (Cryptographically sign and optionally encrypt message) cryptographically signs and optionally encrypts a message.
- “gss_wrap_size_limit()—Determine Largest Message that can be Wrapped” on page 67 (Determine largest message that can be wrapped) determines the largest message that can be processed by the gss_wrap() routine without exceeding the specified output token size.
- “qkrb_build_spnego_init_token()—Build a SPNEGO Initiator Token” on page 69 (Build a SPNEGO initiator token) builds a Simple and Protected GSS-API Negotiation (SPNEGO) Initiator Token and returns the results to the caller.
- “qkrb_build_spnego_target_token()—Build a SPNEGO Target Token” on page 70 (Build a SPNEGO target token) builds a Simple and Protected GSS-API Negotiation (SPNEGO) Target Token and returns the results to the caller.
- “qkrb_free_spnego_init_item()—Release Storage Associated with an Initiator Token Item” on page 73 (Release storage associated with an initiator token item) releases storage associated with a qkrb_spnego_init_item_t object.
- “qkrb_free_spnego_target_item()—Release Storage Associated with a Target Token Item” on page 74 (Release storage associated with a target token item) releases storage associated with a qkrb_spnego_target_item t object.
- “qkrb_parse_spnego_init_token()—Parse a SPNEGO Initiator Token” on page 71 (Parse a SPNEGO initiator token) parses a Simple and Protected GSS-API Negotiation (SPNEGO) Initiator Token and returns the results to the caller.
- “qkrb_parse_spnego_target_token()—Parse a SPNEGO Target Token” on page 72 (Parse a SPNEGO target token) parses a Simple and Protected GSS-API Negotiation (SPNEGO) Target Token and returns the results to the caller.

APIs

These are the APIs for this category.

gss_accept_sec_context()—Accept Security Context

Syntax

```
#include <gssapi.h>

OM_uint32 gss_accept_sec_context (
```

```
OM_uint32 *    minor_status,
gss_ctx_id_t *   context_handle,
gss_cred_id_t    acceptor_cred_handle,
gss_buffer_t   input_token,
gss_channel_bindings_t input_chan_bindings,
gss_name_t *   src_name,
gss_OID *      mech_type,
gss_buffer_t   output_token,
gss_flags_t *    ret_flags,
OM_uint32 *    time_rec,
gss_cred_id_t *    delegated_cred_handle);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_accept_sec_context()** function accepts a security context created by the context initiator.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input/Output)
A context handle for the context. The first time the context acceptor calls the **gss_accept_sec_context()** routine, the context handle value must be set to **GSS_C_NO_CONTEXT**. For subsequent calls to continue setting up the context, the context handle must be the value returned by the previous call to the **gss_accept_sec_context()** routine.

acceptor_cred_handle (Input)
The GSS credential for the identity claimed by the context acceptor. The credential must have been created using either **GSS_C_ACCEPT** or **GSS_C_BOTH**.

input_token (Input)
The token received from the context initiator.

input_chan_bindings (Input)
The bindings describing the communications channel used between the communicating applications. The channel bindings specified by the context acceptor must match the bindings that were specified by the context initiator when the input token was created. Specify **GSS_C_NO_CHANNEL_BINDINGS** if there are no channel bindings.

src_name (Output)
The authenticated name of the context initiator. If the authenticated name is not required, specify **NULL** for this parameter. The returned name is an anonymous internal name if the **GSS_C_ANON_FLAG** is set in the returned flags.

mech_type (Output)
The security mechanism with which the context was established. If the security mechanism type is not required, specify **NULL** for this parameter. The gss_OID value returned for this parameter points to a read-only structure and must not be released by the application. The returned security mechanism is one of the following:

gss_mech_krb5_old Beta Kerberos V5 mechanism
gss_mech_krb5 Kerberos V5 mechanism

output_token (Output)
A token to be returned to the context initiator. If no token is to be passed to the context initiator, the **gss_accept_sec_context()** routine sets the *output_token length* field to zero. Otherwise, the

output_token length and *value* fields are set to nonzero values. The application should release the output token when it is no longer needed by calling the **gss_release_buffer()** routine.

ret_flags (Output)
A bit mask containing independent flags representing services that have been requested by the initiating application. Specify **NULL** for this parameter if the flag values are not required. The following symbolic definitions are provided to test the individual flags and should be logically ANDed with the value of *ret_flags* to test whether the context supports the service option.

GSS_C_ANON_FLAG	Anonymous services are available if this flag is TRUE. The *src_name* parameter returns an anonymous internal name.
GSS_C_CONF_FLAG	Confidentiality services are available if this flag is TRUE.
GSS_C_DELEG_FLAG	Delegated credentials are available if this flag is TRUE.
GSS_C_INTEG_FLAG	Integrity services are available if this flag is TRUE.
GSS_C_MUTUAL_FLAG	Mutual authentication is required if this flag is TRUE.
GSS_C_PROT_READY_FLAG	Protection services, as specified by the **GSS_C_CONF_FLAG** and **GSS_C_INTEG_FLAG**, are available if the accompanying major status is **GSS_S_COMPLETE** or **GSS_S_CONTINUE_NEEDED**. Otherwise, protection services are available only if the accompanying major status is **GSS_S_COMPLETE**.
GSS_C_REPLAY_FLAG	Replayed signed or sealed messages are detected if this flag is TRUE.
GSS_C_SEQUENCE_FLAG	Out-of-sequence signed or sealed messages are detected if this flag is TRUE.

time_rec (Output)
The number of seconds remaining before the context is no longer valid. If the mechanism does not support credential expiration, the return value is **GSS_C_INDEFINITE**. Specify **NULL** for this parameter if the remaining time is not required.

delegated_cred_handle (Output)
The credential handle for delegated credentials received from the context initiator. Specify **NULL** for this parameter if the delegated credentials are not required. A credential handle is returned only if the **GSS_C_DELEG_FLAG** flag is set in the return flags. The returned credential can then be used to initiate a new security context by calling the **gss_init_sec_context()** routine. The returned credential should be released when it is no longer needed by calling the **gss_release_cred()** routine.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_BINDINGS
The *input_token* parameter contains different channel bindings from those specified with the *input_chan_bindings* parameter.

GSS_S_BAD_MECH
The security mechanism used by the context initiator is not available on the acceptor system.

GSS_S_BAD_SIG
The received input token contains an incorrect signature.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTINUE_NEEDED
Control information in the returned output token must be sent to the initiator and a response must be received and passed as the *input_token* argument to a continuation call to the **gss_accept_sec_context()** routine.

GSS_S_CREDENTIALS_EXPIRED
Credentials are no longer valid.

GSS_S_DEFECTIVE_CREDENTIAL
Consistency checks performed on the credential structure referenced by the *verifier_cred_handle* parameter failed.

GSS_S_DEFECTIVE_TOKEN
Consistency checks performed on the input token failed.

GSS_S_DUPLICATE_TOKEN
The token is a duplicate of a token that already has been processed. This is a fatal error during context establishment.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

GSS_S_NO_CRED
No credentials are available or the credentials are valid for context initiation use only.

GSS_S_OLD_TOKEN
The token is too old to be checked for duplication against previous tokens. This is a fatal error during context establishment.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R
Each directory in the path name preceding the keytab file	*X
Keytab file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. The **gss_accept_sec_context()** routine is the second step in establishing a security context between the context initiator and the context acceptor. In the first step, the context initiator calls the **gss_init_sec_context()** routine, which returns a token for the security context. The context initiator then passes this security token to the context acceptor. In the second step, the context acceptor takes the token supplied by the context initiator and calls the **gss_accept_sec_context()** routine to accept the context.

   If the length value in the *output_token* is not zero, the context acceptor must pass the returned token to the context initiator. The context initiator must then call **gss_init_sec_context()** and specify the context identifier returned by the original call to **gss_init_sec_context()**, as well as the output token that was returned by the context acceptor.

To complete the context establishment, one or more reply tokens may be required from the peer application. If so, **gss_accept_sec_context()** returns a status flag of **GSS_S_CONTINUE_NEEDED**, in which case it should be called again when the reply token is received from the peer application, passing the token to **gss_accept_sec_context()** through the *input_token* parameter.

2. The availability of confidentiality services is dependent on the underlying security mechanism and the features that have been installed on the system. The **GSS_C_CONF_FLAG** is returned only if confidentiality services are available on both the local and remote systems. If confidentiality services are available on the remote system but not on the local system, an error is returned by the **gss_unwrap()** routine if an encrypted message is received (that is, confidentiality was requested on the call to the **gss_wrap()** routine on the remote system).
3. Whenever the **GSS_S_CONTINUE_NEEDED** status flag is set, the context is not fully established and the following restrictions apply to the output parameters:

   - The value returned by the *time_rec* parameter is undefined.
   - Unless the accompanying *ret_flags* parameter contains the bit **GSS_C_PROT_READY_FLAG**, indicating that per-message services may be applied in advance of a successful completion status, the value returned by the *mech_type* parameter may be undefined until the routine returns a major status of **GSS_S_COMPLETE**.
   - The values of the **GSS_C_DELEG_FLAG**, **GSS_C_MUTUAL_FLAG**, **GSS_C_REPLAY_FLAG**, **GSS_C_SEQUENCE_FLAG**, **GSS_C_CONF_FLAG**, **GSS_C_INTEG_FLAG**, and **GSS_C_ANON_FLAG** bits returned by the *ret_flags* parameter contain the values that the implementation expects would be valid if context establishment were to succeed.
   - The value of the **GSS_C_PROT_READY_FLAG** bit returned by the *ret_flags* parameter indicates the actual state at the time **gss_accept_sec_context()** returns, whether or not the context is fully established.

4. Kerberos mechanism

   - The **gss_accept_sec_context()** routine needs a key to decrypt the token provided by the context initiator. The token contains the clear text principal name of the context acceptor. This name identifies the key that the context initiator used to encrypt the token. The default key table is used to obtain the key for the indicated principal. The **KRB5_KTNAME** environment variable can be set to use a different key table.
   - The context expiration time is obtained from the service ticket that was obtained by the context initiator as part of the **gss_init_sec_context()** processing.
   - When delegation is used, the forwarded Kerberos credentials are stored in a new Kerberos credentials cache that will be associated with the GSS credential returned for the *delegated_cred_handle* parameter. This GSS credential can then be used to initiate new security contexts on behalf of the original context initiator.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_acquire_cred()—Acquire GSS Credential

Syntax

```
#include <gssapi.h>

OM_uint32 gss_acquire_cred(
    OM_uint32 *    minor_status,
    gss_name_t     desired_name,
```

```
OM_uint32      time_req,
gss_OID_set    desired_mechs,
gss_cred_usage_t   cred_usage,
gss_cred_id_t *    output_cred_handle,
gss_OID_set *    actual_mechs,
OM_uint32 *    time_rec);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_acquire_cred()** function allows an application to acquire a GSS credential. The application can then use the credential with the **gss_init_sec_context()** and **gss_accept_sec_context()** routines.

Parameters

minor_status (Output)
A status code from the security mechanism.

desired_name (Input)
The principal name to be used for the credential. Specify **GSS_C_NO_NAME** for this parameter to use the name obtained from the default login context.

time_req (Input)
The number of seconds that the credential remains valid. Specify **GSS_C_INDEFINITE** to request the maximum credential lifetime. Specify zero for the default lifetime of 2 hours. The actual credential lifetime is limited by the lifetime of the underlying ticket-granting ticket for **GSS_C_INITIATE** and **GSS_C_BOTH** credentials.

desired_mechs (Input)
The desired security mechanisms for use with the credential. Mechanisms that are not available on the local system are ignored. The actual mechanisms that can be used with the credential are returned in the *actual_mechs* parameter. Specify **GSS_C_NO_OID_SET** for this parameter to use the default mechanism of **gss_mech_krb5**.

The following security mechanisms are supported:

gss_mech_krb5_old	Beta Kerberos V5 mechanism
gss_mech_krb5	Kerberos V5 mechanism

cred_usage (Input)
The desired credential usage as follows:

GSS_C_ACCEPT	The credential can be used only to accept security contexts.
GSS_C_BOTH	The credential can be used to both initiate and accept security contexts.
GSS_C_INITIATE	The credential can be used only to initiate security contexts.

output_cred_handle (Output)
The handle for the GSS credential.

actual_mechs (Output)
The set of mechanism identifiers for which the credential is valid. If the actual mechanisms are not required, specify **NULL** for this parameter. The gss_OID_set returned for this parameter should be released by calling the **gss_release_oid_set()** routine when it is no longer needed.

time_rec (Output)
The number of seconds for which the credential will remain valid. If the time remaining is not required, specify **NULL** for this parameter.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_MECH
None of the requested mechanisms are supported by the local system.

GSS_S_BAD_NAME
The name specified for the *desired_name* parameter is not valid.

GSS_S_BAD_NAMETYPE
The name specified for the *desired_name* parameter is not supported by the applicable underlying GSS mechanisms.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
No credentials are available or the credentials are valid for context initiation use only.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R
Each directory preceding the credential cache file if **GSS_C_INITIATE** or **GSS_C_BOTH** is specified for credential usage	*X
Credential cache file	*RW
Each directory preceding the keytab file if **GSS_C_ACCEPT** or **GSS_C_BOTH** is specified for credential usage	*X
Keytab file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. If **GSS_C_INITIATE** or **GSS_C_BOTH** is specified for the credential usage, the application must have a valid ticket-granting ticket in the default credentials cache and the ticket must not expire for at least 10 minutes. The **gss_acquire_cred()** routine uses this ticket-granting ticket to create the GSS credential. The principal specified by the *desired_name* parameter must match the principal obtained from the

credentials cache or must be specified as **GSS_C_NO_NAME**. The **KRB5CCNAME** environment variable is used to identify the credentials cache used by the Kerberos security mechanism.

2. If **GSS_C_ACCEPT** or **GSS_C_BOTH** is specified for the credential usage, the principal specified by the *desired_name* parameter must be defined in a key table. The **KRB5_KTNAME** environment variable can be used to set the key table used by the Kerberos security mechanism.

API introduced: V5R1

---

gss_add_cred()—Add Credential Element to Existing GSS Credential

Syntax

```
#include <gssapi.h>

OM_uint32 gss_add_cred(
    OM_uint32 *    minor_status,
    gss_cred_id_t    input_cred_handle,
    gss_name_t     desired_name,
    gss_OID      mech_type,
    gss_cred_usage_t   cred_usage,
    OM_uint32      init_time_req,
    OM_uint32      accept_time_req,
    gss_cred_id_t *    output_cred_handle,
    gss_OID_set *    actual_mechs,
    OM_uint32 *    init_time_rec,
    OM_uint32 *    accept_time_rec);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_add_cred()** function adds a credential element to an existing GSS credential. The credential must not already contain an element for the mechanism. A GSS credential must contain an element for each mechanism that will be used for contexts that are initiated or accepted using the credential.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_cred_handle (Input)
The GSS credential that is to be modified. Specify **GSS_C_NO_CREDENTIAL** to modify the default GSS credential.

desired_name (Input)
The principal name to be used for the credential.

mech_type (Input)
The mechanism element to be added to the credential. The credential must not already contain an element for this mechanism.

The following security mechanisms are supported:

gss_mech_krb5_old Beta Kerberos V5 mechanism
gss_mech_krb5 Kerberos V5 mechanism

cred_usage (Input)
The desired credential usage as follows:

GSS_C_ACCEPT The credential can be used only to accept security contexts.
GSS_C_BOTH The credential can be used to both initiate and accept security contexts.
GSS_C_INITIATE The credential can be used only to initiate security contexts.

init_time_req (Input)
The number of seconds the credential remains valid for initiating contexts. The i5/OS® implementation of GSS does not support separate initiate and accept expiration times. The actual expiration time will be the smaller of the initiate and accept times. Specify zero to request the default lifetime of 2 hours. Specify **GSS_C_INDEFINITE** to request the maximum lifetime.

accept_time_req (Input)
The number of seconds the credential remains valid for accepting contexts. The i5/OS implementation of GSS does not support separate initiate and accept expiration times. The actual expiration time will be the smaller of the initiate and accept times. Specify zero to request the default lifetime of 2 hours. Specify **GSS_C_INDEFINITE** to request the maximum lifetime.

output_cred_handle (Output)
The credential handle for the updated credential. If **NULL** is specified for this parameter, the new credential element is added to the input credential. Otherwise, a new credential is created from the input credential and contains all of the credential elements of the input credential plus the new credential element. **NULL** may not be specified for this parameter if **GSS_C_NO_CREDENTIAL** is specified for the input credential.

actual_mechs (Output)
The total set of mechanisms supported by the GSS credential. Specify **NULL** for this parameter if the actual mechanisms are not required. The gss_OID_set returned for this parameter should be released by calling the **gss_release_oid_set()** routine when it is no longer needed.

init_time_rec (Output)
The initiate expiration time in seconds. Specify **NULL** for this parameter if the initiate time is not required.

accept_time_rec (Output)
The accept expiration time in seconds. Specify **NULL** for this parameter if the accept time is not required.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_MECH
The specified mechanism is not supported.

GSS_S_BAD_NAME
The name specified for the *desired_name* parameter is not valid.

GSS_S_BAD_NAMETYPE
The name specified for the *desired_name* parameter is not supported by the applicable underlying GSS mechanisms.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_DUPLICATE_ELEMENT
The credential already contains an element for the specified mechanism.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The referenced credential does not exist.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. The **gss_add_cred()** routine performs the same functionas the **gss_acquire_cred()** routine for a single mechanism.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_add_oid_set_member()—Add OID to an OID Set

Syntax

```
#include <gssapi.h>

OM_uint32 gss_add_oid_set_member(
    OM_uint32 *    minor_status,
    gss_OID      input_oid,
    gss_OID_set *    oid_set);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_add_oid_set_member()** function adds a new OID to an existing OID set.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_oid (Input)
The OID to add to the OID set.

oid_set (Input/Output)
The OID set. The gss_OID array referenced by the *elements* field of the gss_OID_set will be reallocated to hold the new OID. The application should call the **gss_release_oid_set()** routine to release the OID set when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE	The routine completed successfully.
GSS_S_FAILURE	The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. You can create an empty OID set by calling the **gss_create_empty_oid_set()** routine. The **gss_add_oid_set_member()** routine makes a copy of the input OID, so any future changes to the input OID will have no effect on the copy in the OID set.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_canonicalize_name()—Reduce GSS Internal Name to Mechanism Name

Syntax

```
#include <gssapi.h>

OM_uint32 gss_canonicalize_name(
    OM_uint32 *    minor_status,
    gss_name_t     input_name,
    gss_OID      mech_type,
    gss_name_t *   output_name);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_canonicalize_name()** routine takes a GSS internal name that contains multiple internal representations and returns a new GSS internal name with a single name representation that corresponds to the specified security mechanism. A name that represents a single security mechanism is called a **mechanism name**.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_name (Input)
The name to be processed. An error is returned if **GSS_C_NO_NAME** is specified for this parameter.

mech_type (Input)
The security mechanism to be used.

The following security mechanisms are supported:

gss_mech_krb5_old Beta Kerberos V5 mechanism
gss_mech_krb5 Kerberos V5 mechanism

output_name (Output)
The mechanism name. The gss_name_t returned by this parameter should be released by calling the **gss_release_name()** function when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_MECH
The specified mechanism is not supported.

GSS_S_BAD_NAME
The input name is not valid.

GSS_S_BAD_NAMETYPE
The input name does not contain an element for the mechanism.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_compare_name()—Compare Two Internal GSS Names

Syntax

```
#include <gssapi.h>

OM_uint32 gss_compare_name(
    OM_uint32 *    minor_status,
    gss_name_t     name1,
    gss_name_t     name2,
    int *      name_equal);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_compare_name()** function allows an application to compare two internal names to determine whether they refer to the same object. The two names must have an internal representation format in common to be comparable. The names are considered not equal if either name denotes an anonymous principal.

Parameters

minor_status (Output)
A status code from the security mechanism.

name1 (Input)
The first internal name.

name2 (Input)
The second internal name.

name_equal (Output)
Returns 1 if the names refer to the same object and 0 otherwise.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_NAME
One of the input names is not valid.

GSS_S_BAD_NAMETYPE
The two name types are not comparable. The names must have an internal representation in common to be comparable.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

---

gss_context_time()—Get Number of Seconds Security Context Remains Valid

Syntax

```
#include <gssapi.h>

OM_uint32 gss_context_time(
    OM_uint32 *    minor_status,
    gss_ctx_id_t   context_handle,
    OM_uint32 *    time_rec);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_context_time()** function checks the specified security context and returns the number of seconds that the context remains valid. The returned value is **GSS_C_INDEFINITE** if the context does not have an expiration time. The Kerberos security mechanism does support context expiration and returns the time remaining before the underlying service ticket expires.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The context to be checked.

time_rec (Output)
The number of seconds that the context remains valid.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The referenced context has expired.

GSS_S_CREDENTIALS_EXPIRED
The credentials associated with the referenced context have expired.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The referenced context does not exist.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_create_empty_oid_set()—Create Empty OID Set

Syntax

```
#include <gssapi.h>

OM_uint32 gss_create_empty_oid_set(
    OM_uint32 *    minor_status,
    gss_OID_set *    oid_set);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_create_empty_oid_set()** function creates a new, empty OID set. Members can be added to the OID set by calling the **gss_add_oid_set_member()** routine. The OID set should be released when it is no longer needed by calling the **gss_release_oid_set()** routine.

Parameters

minor_status (Output)
A status code from the security mechanism.

oid_set (Output)
The OID set created by this routine. The application should call the **gss_release_oid_set()** routine to release the OID set when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_delete_sec_context()—Delete Security Context

Syntax

```
#include <gssapi.h>

OM_uint32 gss_delete_sec_context (
    OM_uint32 *    minor_status,
    gss_ctx_id_t *   context_handle,
    gss_buffer_t   output_token);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_delete_sec_context()** function deletes one end of a security context. It also deletes the local data structures associated with the security context. When it deletes the context, the routine can generate a token. The application must then pass this token to the partner application. The partner application calls the **gss_process_context_token()** routine to process the token and complete the process of deleting the security context.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input/Output)
The context to be deleted. Upon successful completion, the *context_handle* value is set to **GSS_C_NO_CONTEXT**.

output_token (Output)
A token to be sent to the partner application. The partner application then passes this token to the **gss_process_context_token()** routine to delete the other end of the security context. The **gss_delete_sec_context()** routine sets the *output_token length* field to zero if no token needs to be sent to the partner application.

GSS_C_NO_BUFFER may be specified for the *output_token* parameter. In this case, no token is returned by the **gss_delete_sec_context()** routine. Both of the communicating applications must call **gss_delete_sec_context()** to delete both ends of the security context.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. This call can be made by either peer in a security context to flush context-specific information. Both communicating applications must call the **gss_delete_sec_context()** routine if **GSS_C_NO_BUFFER** is specified for the *output_token* parameter.
2. The *context_handle* may not be used for additional security services once the **gss_delete_sec_context()** routine has completed successfully.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_display_name()—Get Textual Representation of Internal GSS Name

Syntax

```
#include <gssapi.h>

OM_uint32 gss_display_name(
    OM_uint32 *    minor_status,
    gss_name_t     input_name,
    gss_buffer_t   output_name_buffer,
    gss_OID *      output_name_type);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_display_name()** function returns the textual representation of an opaque internal name. The syntax of the text representation is determined by the mechanism that was used to convert the name.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_name (Input)
The internal name to be converted to a text string.

output_name_buffer (Output)
Return buffer for the character string.

output_name_type (Output)
The name type corresponding to the returned character string. The gss_OID value returned for this parameter points to read-only storage and must not be released by the application.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_NAME
The provided name is not valid.

GSS_S_BAD_NAMETYPE
The internal name provided does not have an internal representation for any of the supported mechanisms.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. Kerberos names are formatted as **principal-name@realm-name**.

   Not every coded character set identifier (CCSID) contains the '@' character; however, alternative CCSID values often are available. For example, instead of using Greece 423, run the job with a default CCSID of 875.

API introduced: V5R1

gss_display_status()—Get Textual Representation of GSS Status Code or Mechanism Code

Syntax

```
#include <gssapi.h>

OM_uint32 gss_display_status(
    OM_uint32 *  minor_status,
    OM_uint32   status_value,
    int   status_type,
    gss_OID    mech_type,
    gss_msg_ctx_t *  message_context,
    gss_buffer_t  status_string);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_display_status()** function provides an application with a textual representation of a GSS or mechanism status code. The returned message can then be displayed to the user or written to a log file.

Parameters

minor_status (Output)
A status code from the security mechanism.

status_value (Input)
The status value to be converted. A status value of zero is not valid and causes the **gss_display_status()** routine to return a major status of **GSS_S_BAD_STATUS** to the application.

status_type (Input)
The status value type. The status value type must be one of the following:

GSS_C_GSS_CODE	GSS major status code
GSS_C_MECH_CODE	Mechanism minor status code

mech_type (Input)
The security mechanism associated with a minor status code. This parameter is used only when converting a minor status code.

message_context (Input/Output)
Whether the status code has multiples messages to be processed. The first time an application calls **gss_display_status()**, the *message_context* parameter must be initialized to zero. The **gss_display_status()** routine returns the first message and sets the *message_context* parameter to a nonzero value if more messages are available. The application then continues to call the **gss_display_status()** routine to obtain the additional messages until the *message_context* value is zero upon return from the **gss_display_status()** routine.

status_string (Output)
The text message for the status value.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_MECH
The mechanism specified by the *mech_type* parameter is not supported.

GSS_S_BAD_STATUS
The value of the *status_type* parameter is not **GSS_C_GSS_CODE** or **GSS_C_MECH_CODE** or the value of the *status_value* parameter is not a valid status code.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. The *message_context* parameter indicates which error message should be returned when a status code has multiple messages. The first time an application calls the **gss_display_status()** routine, it must initialize the *message_context* value to zero. The **gss_display_status()** routine then returns the first message for the status code and sets *message_context* to a nonzero value if there are additional messages available. The application can then continue to call **gss_display_status()** until the *message_context* value is zero upon return.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_duplicate_name()—Create Duplicate GSS Internal Name

Syntax

```
#include <gssapi.h>

OM_uint32 gss_duplicate_name(
    OM_uint32 *    minor_status,
    gss_name_t     input_name,
    gss_name_t *   output_name);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_duplicate_name()** function creates a duplicate of a GSS internal name.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_name (Input)
The name to be duplicated. An error is returned if **GSS_C_NO_NAME** is specified for this parameter.

output_name (Output)
The new GSS internal name. The gss_name_t returned for this parameter should be released by calling the **gss_release_name()** function when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_NAME
The input name is not valid.

GSS_S_BAD_NAMETYPE
The input name type is not supported.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_export_cred()—Export GSS Credential

Syntax

```
#include <krb5.h>

krb5_error_code gss_export_cred (
   OM_uint32 *                          minor_status,
   gss_cred_id_t                        cred_handle,
   gss_buffer_t                         cred_token)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_export_cred()** routine creates a credential token for a GSS-API credential. This credential token can then be given to another process on the same system or on a different system. This second process calls **gss_import_cred()** to create a GSS-API credential from the credential token. In order to use the credential on a different system, the security mechanism must allow the credential to be used from any system. In the case of the Kerberos security mechanism, this means the Kerberos ticket must not contain a client address list.

A credential can be exported only if it is an initiate credential (GSS_C_INITIATE was specified when the credential was created). The major status will be set to GSS_S_NO_CRED if the credential is not an initiate credential. The credential remains available upon completion of the export operation and can be used in subsequent GSS-API operations. The credential token created by one implementation of GSS-API cannot be used with a different implementation of GSS-API.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

cred_handle (Input/Output)
The credential handle of the GSS-API credential to be used to create t he credential token. The credential must be an initiate credential.

cred_token (Output)
The credential token returned. The storage for the token should be released when it is no longer needed by calling the **gss_release_buffer()** routine.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The supplied credential handle does not refer to a valid credential.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

gss_export_name()—Create Opaque Token for a Mechanism Name

Syntax

```
#include <krb5.h>

krb5_error_code gss_export_name (
   OM_uint32 *                        minor_status,
   gss_name_t                         input_name,
   gss_buffer_t                  exported_name)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_export_name()** routine creates an opaque token for a mechanism name.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

input_name (Input)
The GSS-API name to be exported. This must represent a mechanism name.

exported_name (Output)
The token returned that represents the GSS-API name. The **gss_release_buffer()** routine should be called to release the token when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NAME_NOT_MN
The supplied name is not a mechanism name. Use the **gss_canonicalize_name()** routine to convert an internal name to a mechanism name.

GSS_S_BAD_NAMETYPE
The input name is not supported by the current GSS-API Implementation.

GSS_S_BAD_NAME
The input name is not valid.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

The **gss_canonicalize_name()** routine will convert a GSS-API internal name with multiple mechanism representations to a mechanism name. The **gss_canonicalize_name()** and **gss_export_name()** calls enable callers to acquire and process exported name objects, canonicalized and translated in accordance with the procedures of a particular GSS-API mechanism. Exported name objects can, in turn, be input to **gss_import_name()**, yielding equivalent mechanism names. These facilities are designed specifically to enable efficient storage and comparison of names (for example, for use in access control lists).

API introduced: V5R2

---

gss_export_sec_context()—Export Security Context

Syntax

```
#include <krb5.h>

krb5_error_code gss_export_sec_context (
   OM_uint32 *                        minor_status,
   gss_ctx_id_t *                     context_handle,
   gss_buffer_t                       context_token)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_export_sec_context()** routine creates a context token for a GSS-API security context. This context token can then be given to another process on the same system. This second process calls **gss_import_sec_context()** to create a GSS-API security context from the context token. Upon successful completion of **gss_export_sec_context()**, the security context is no longer available for use by the current process. The security context token created by one implementation of GSS-API cannot be used with a different implementation of GSS-API.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

context_handle (Input/Output)
The context handle of the GSS-API security context to be used to create the security context token. The context handle will be set to GSS_C_NO_CONTEXT upon successful completion.

context_token (Output)
The security context token returned. The storage for the token should be released when it is no longer needed by calling the **gss_release_buffer()** routine.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The supplied context handle does not refer to a valid context.

GSS_S_CONTEXT_EXPIRED
The supplied context is no longer valid.

GSS_S_UNAVAILABLE
Security context cannot be exported.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_get_mic()—Generate Cryptographic Signature for Message

Syntax

```
#include <gssapi.h>

OM_uint32 gss_get_mic(
    OM_uint32 *    minor_status,
    gss_ctx_id_t   context_handle,
    gss_qop_t      qop_req,
    gss_buffer_t     input_message
    gss_buffer_t     output_token);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_get_mic()** function generates a cryptographic signature for a message and returns this signature in a token that can be sent to a partner application. The partner application then calls the **gss_verify_mic()** routine to validate the signature.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The context to be associated with the message when it is sent to the partner application.

qop_req (Input)
The requested quality of protection for the message. Specify **GSS_C_QOP_DEFAULT** to use the default quality of protection as defined by the selected security mechanism.

The Kerberos security mechanism supports three quality of protection levels as follows (in decreasing order or speed):

GSS_KRB5_INTEG_C_QOP_MD5	Truncated MD5
GSS_KRB5_INTEG_C_QOP_DES_MD5	DES_MAC of an MD5 hash (default)
GSS_KRB5_INTEG_C_QOP_DES_MAC	Normal DES_MAC algorithm

input_message (Input)
The message for which a signature is to be generated.

output_token (Output)
A token containing the message signature. The message and this token are then sent to the partner application, which calls the **gss_verify_mic()** function to verify the authenticity of the message.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_QOP
The requested quality of protection value is not valid.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The referenced context has expired.

GSS_S_CREDENTIALS_EXPIRED
The credentials associated with the referenced context have expired.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_import_cred()—Import GSS Credential

Syntax

```
#include <krb5.h>

krb5_error_code gss_import_cred (
   OM_uint32 *                         minor_status,
   gss_buffer_t                        cred_token,
   gss_ctx_id_t *                      cred_handle)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_import_cred()** routine accepts a credential token created by the **gss_export_cred()** routine and creates a GSS-API credential.

The **gss_release_cred()** routine should be called to release the GSS-API credential when it is no longer needed. The credential token created by one implementation of GSS-API cannot be used with a different implementation of GSS-API.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

cred_token (Input)
The credential token created by the **gss_export_cred()** routine.

cred_handle (Output)
The credential handle returned for the GSS-API credential created from the credential token. The **gss_release_cred()** routine should be called to release the credential when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_DEFECTIVE_TOKEN
The supplied credential token is not valid.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_import_name()—Convert Printable Name to GSS Internal Format

Syntax

```
#include <gssapi.h>

OM_uint32 gss_import_name(
    OM_uint32 *    minor_status,
    gss_buffer_t   input_name_buffer,
    gss_OID      input_name_type,
    gss_name_t *   output_name);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_import_name()** function converts a printable name to the GSS internal format. The gss_name_t object created by this routine can then be used as input to other GSS routines. The gss_name_t object created by the **gss_import_name()** routine contains an internal representation for each of the supported security mechanisms.

Not every coded character set identifier (CCSID) contains the '@' character; however, alternative CCSID values often are available. For example, instead of using Greece 423, run the job with a default CCSID of 875.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_name_buffer (Input)
The buffer containing the name to convert.

input_name_type (Input)
The object identifier for the type of printable name.

The following name types are supported:

GSS_C_NO_OID	The default name type. For the i5/OS® implementation of GSS, the default is **GSS_C_NT_USER_NAME**.
GSS_C_NT_USER_NAME	For the Kerberos mechanism, this is assumed to be the name of a Kerberos principal in the format **principal@realm**.
GSS_C_NT_HOSTBASED_SERVICE	A service that is related to a particular host. For the Kerberos mechanism, the service name is specified as **service@host**. The service name is mapped to the principal **service/primary-host@realm** using the **krb5_sname_to_principal()** function. The primary host name must be associated with a Kerberos realm to map the service name to the proper principal.
GSS_C_NT_HOSTBASED_SERVICE_X	A service that is related to a particular host. This is the same as **GSS_C_NT_HOSTBASED_SERVICE** and should not be used by new applications.
gss_nt_krb5_name	A Kerberos name in the format **principal@realm**. This name type is valid only for the Kerberos mechanism.
gss_nt_krb5_principal	A krb5_principal created by the **krb5_parse_name()** routine. This name type is valid only for the Kerberos mechanism.

output_name (Output)
The name in the GSS internal format. The internal format contains an internal representation for each of the supported security mechanisms.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_NAME
The input name is not formatted properly or is not valid.

GSS_S_BAD_NAMETYPE
The name type specified by the *input_name_type* parameter is not valid.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_import_sec_context()—Import Security Context

Syntax

```
#include <krb5.h>

krb5_error_code gss_import_sec_context (
   OM_uint32 *                        minor_status,
   gss_buffer_t                       context_token,
   gss_ctx_id_t *                     context_handle)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_import_sec_context()** routine accepts a security context token created by the strong>gss_export_sec_context() routine and creates a GSS-API security context. Since the security context contains message sequencing information, it is usually not feasible to create multiple security contexts from a single context token.

The **gss_delete_sec_context()** routine should be called to delete the GSS-API security context when it is no longer needed. The security context token created by one implementation of GSS-API cannot be used with a different implementation of GSS-API.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

context_token (Input)
The security context token created by the **gss_export_sec_context()** routine.

context_handle (Output)
The context handle returned for the security context created from the context token. The **gss_delete_sec_context()** routine should be called to delete the security context when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_DEFECTIVE_TOKEN
The supplied credential token is not valid.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_indicate_mechs()—Determine Available Security Mechanisms

Syntax

```
#include <gssapi.h>

OM_uint32 gss_indicate_mechs(
    OM_uint32 *    minor_status,
    gss_OID_set *    mech_set);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_indicate_mechs()** function allows an application to determine which security mechanisms are available on the local system.

Parameters

minor_status (Output)
A status code from the security mechanism.

mech_set (Output)
The set of supported security mechanisms. The application should release the gss_OID_set returned for this parameter by calling the **gss_release_oid_set()** routine.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_init_sec_context()—Initiate Security Context

Syntax

```
#include <gssapi.h>

OM_uint32 gss_init_sec_context (
    OM_uint32 *    minor_status,
    gss_cred_id_t    cred_handle,
    gss_ctx_id_t *   context_handle,
    gss_name_t     target_name,
    gss_OID      mech_type,
    gss_flags_t    req_flags,
    OM_uint32      time_req,
    gss_channel_bindings_t input_chan_bindings,
    gss_buffer_t   input_token,
    gss_OID *      actual_mech_type,
    gss_buffer_t   output_token,
    gss_flags_t *    ret_flags,
    OM_uint32 *    time_rec);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_init_sec_context()** function initiates a security context for use by two communicating applications.

Parameters

minor_status (Output)
A status code from the security mechanism.

cred_handle (Input)
The credential handle of the GSS credential to be used to initiate the security context. The specified credential must have been created using either **GSS_C_INITIATE** or **GSS_C_BOTH**. Specify **GSS_C_NO_CREDENTIAL** to use the default credential obtained from the current login context.

context_handle (Input/Output)
The context handle for the context. The first time the context initiator calls the **gss_init_sec_context()** routine, the context handle must be set to **GSS_C_NO_CONTEXT**. For subsequent calls to continue setting up the context, the context handle must be the value returned by the previous call to the **gss_init_sec_context()** routine.

target_name (Input)
The name of the context acceptor. This must be a Kerberos service name if delegation is requested for the Kerberos security mechanism. Otherwise, it can be any principal defined in the security registry, subject to registry policy rules.

mech_type (Input)
The desired security mechanism as follows:

gss_mech_krb5_old	Beta Kerberos V5 mechanism
gss_mech_krb5	Kerberos V5 mechanism
GSS_C_NO_OID	Default mechanism. For the i5/OS® implementation of GSS, this is the Kerberos V5 mechanism.

req_flags (Input)
A bit mask containing independent flags representing requested GSS services. GSS does not guarantee that a requested service will be available on all systems. The application should check the *ret_flags* parameter to determine which of the requested services are actually provided for the context. The following symbolic definitions are provided to correspond to each flag. The symbolic names should be logically ORed to form the bit mask value.

GSS_C_ANON_FLAG	Request initiator anonymity. This flag is ignored in the current GSS implementation since Kerberos mechanism does not support initiator anonymity.
GSS_C_DELEG_FLAG	Request delegated credentials for use by the context acceptor.
GSS_C_MUTUAL_FLAG	Request mutual authentication to validate the identity of the context acceptor.
GSS_C_REPLAY_FLAG	Request message replay detection for signed or sealed messages.
GSS_C_SEQUENCE_FLAG	Request message sequence checking for signed or sealed messages.

time_req (Input)
The desired number of seconds the security context remains valid. Specify zero for the default lifetime of 2 hours. Specify **GSS_C_INDEFINITE** to request the maximum lifetime.

input_chan_bindings (Input)
The bindings describing the communications channel that is used between the communicating

applications. The channel bindings information is placed in the output token that is generated by the **gss_init_sec_context()** routine and is validated by the **gss_accept_sec_context()** routine. Specify **GSS_C_NO_CHANNEL_BINDINGS** if there are no channel bindings.

input_token (Input)
The token received from the context acceptor. **GSS_C_NO_BUFFER** should be specified if this is the first call to the **gss_init_sec_context()** routine.

actual_mech_type (Output)
The security mechanism that is used with the context. The gss_OID value returned for this parameter points to read-only storage and must not be released by the application.

output_token (Output)
A token to be sent to the context acceptor. If no token is to be sent to the context acceptor, the **gss_init_sec_context()** routine sets the *output_token length* field to zero. Otherwise, the *output_token length* and *value* fields are set. The application should release the output token when it is no longer needed by calling the **gss_release_buffer()** routine.

ret_flags (Output)
A bit mask containing independent flags indicating which GSS services are available for the context. The following symbolic definitions are provided to test the individual flags and should be logically ANDed with the value of *ret_flags* to test whether the context supports the service options:

GSS_C_ANON_FLAG	The initiator identity will not be provided to the context acceptor.
GSS_C_CONF_FLAG	Message confidentiality services are available.
GSS_C_DELEG_FLAG	Delegated credentials will be available to the context acceptor.
GSS_C_INTEG_FLAG	Message integrity services are available.
GSS_C_MUTUAL_FLAG	Mutual authentication will be performed. The **gss_accept_sec_context()** routine will generate an output token which the context acceptor must return to the context initiator to complete the security context setup.
GSS_C_PROT_READY_FLAG	Protection services, as specified by the states of the **GSS_C_CONF_FLAG** and **GSS_C_INTEG_FLAG**, are available for use if the accompanying major status return value is **GSS_S_COMPLETE** or **GSS_S_CONTINUE_NEEDED**. Otherwise, protection services are available for use only if the accompanying major status return value is **GSS_S_COMPLETE**.
GSS_C_REPLAY_FLAG	Message replay detection will be performed.
GSS_C_SEQUENCE_FLAG	Message sequence checking will be performed.

time_rec (Output)
The number of seconds for which the context will remain valid. If the mechanism does not support context expiration, the return value will be **GSS_C_INDEFINITE**. Specify NULL for this parameter if the context expiration time is not required.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_BAD_BINDINGS
The channel bindings are not valid.

GSS_S_BAD_MECH
The request security mechanism is not supported.

GSS_S_BAD_NAME
The *target_name* parameter is not valid.

GSS_S_BAD_SIG
The input token contains an incorrect integrity check value.

GSS_S_CONTINUE_NEEDED
To complete the context, the **gss_init_sec_context()** routine must be called again with a token created by the **gss_accept_sec_context()** routine.

GSS_S_CREDENTIALS_EXPIRED
The supplied credentials are no longer valid.

GSS_S_DEFECTIVE_CREDENTIAL
Consistency checks performed on the credential failed.

GSS_S_DEFECTIVE_TOKEN
Consistency checks performed on the input token failed.

GSS_S_DUPLICATE_TOKEN
The token is a duplicate of a token that has already been processed.

GSS_S_NO_CONTEXT
The context handle provided by the caller does not refer to a valid security context.

GSS_S_NO_CRED
The supplied credential handle does not refer to a valid credential, the supplied credential is not valid for context initiation, or there are no default credentials available.

GSS_S_OLD_TOKEN
The token is too old to be checked for duplication against previous tokens which have already been processed.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R
Each directory in the path name preceding the credential cache file	*X
Credential cache file	*RW

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. The **gss_init_sec_context()** routine is the first step in the establishment of a security context between the context initiator and the context acceptor. To ensure the portability of the application, use the default credential by specifying **GSS_C_NO_CREDENTIAL** for the *cred_handle* parameter.

The first time the application calls the **gss_init_sec_context()** routine, the *input_token* parameter should either be specified as **GSS_C_NO_BUFFER** or the buffer length field should be set to zero. If no token needs to be sent to the context acceptor, the **gss_init_sec_context()** routine sets the *output_token length* field to zero.

To finish establishing the context, the calling application can require one or more tokens from the context acceptor. If the application requires reply tokens, the **gss_init_sec_context()** routine returns **GSS_S_CONTINUE_NEEDED** in the supplementary information portion of the major status value. The application must call the **gss_init_sec_context()** routine again when it receives the reply token from the context acceptor and pass the token using the *input_token* parameter. When calling the **gss_init_sec_context()** routine to continue processing a context, the same request values must be used as for the initial call.

2. The availability of confidentiality services is dependent on the underlying security mechanism and the features that have been installed on the system. The **GSS_C_CONF_FLAG** is returned only if confidentiality services are available on the local system. This does not guarantee, however, that confidentiality services are also available on the remote system. If confidentiality services are available on the local system but not on the remote system, an error is returned by the **gss_unwrap()** routine on the remote system if an encrypted message is received (that is, confidentiality was requested on the call to the **gss_wrap()** routine on the local system).
3. Whenever the routine returns a major status that includes the value **GSS_S_CONTINUE_NEEDED**, the context is not fully established and the following restrictions apply to the output parameters:

   - The value returned by the *time_rec* parameter is undefined.
   - Unless the accompanying *ret_flags* parameter contains the bit **GSS_C_PROT_READY_FLAG**, indicating that per-message services may be applied in advance of a successful completion status, the value returned by the *actual_mech_type* parameter is undefined until the routine returns a major status value of **GSS_S_COMPLETE**.
   - The values of the **GSS_C_DELEG_FLAG**, **GSS_C_MUTUAL_FLAG**, **GSS_C_REPLAY_FLAG**, **GSS_C_SEQUENCE_FLAG**, **GSS_C_CONF_FLAG**, **GSS_C_INTEG_FLAG**, and **GSS_C_ANON_FLAG** bits returned by the *ret_flags* parameter contain the values that would be returned if the context establishment were to succeed. In particular, if the application has requested a service such as delegation or anonymous authentication through the *req_flags* parameter and such a service is unavailable from the underlying mechanism, **gss_init_sec_context()** generates a token that does not provide the service and indicates through the *ret_flags* parameter that the service is not supported. The application may choose to abort the context establishment by calling **gss_delete_sec_context()** or it may choose to transmit the token and continue context establishment.
   - The value of the **GSS_C_PROT_READY_FLAG** bit returned by the *ret_flags* parameter indicates the actual state at the time **gss_accept_init_context()** returns, whether or not the context is fully established.
4. Kerberos mechanism
   - To use delegation, the target principal name must be a service name. A service name is created by calling the **gss_import_name()** routine with the name type specified as **gss_nt_service_name** (object identifier {1 2 840 113554 1 2 1 4}). The service name is specified as ″name@host″ and results in a Kerberos principal of ″name/host@host-realm″. The local host name is used if no host is specified. If a host name alias is specified, the primary host name returned by the domain name service is used when constructing the principal name. The target principal name is not required to be a service name if the ticket-granting ticket does not contain a client address list. You can obtain a ticket-granting ticket without a client address list by specifying the **-a** option on the **kinit** command. Otherwise, the service name must correctly identify the host on which the target service is running.
   - The requested context lifetime is used to specify the end time when obtaining a Kerberos service ticket to the target application. The actual context lifetime is then set to the lifetime of the ticket, which may be less than the requested lifetime as determined by the registry policy.

- If delegation is requested, the ticket-granting-ticket contained in the login context must allow forwardable tickets. If the ticket-granting ticket is not forwardable, the **gss_init_sec_context()** request will be successful but the **GSS_C_DELEG_FLAG** will not be set in the returned flags. In addition, the service ticket obtained for the target principal must allow delegation. If the target system is not enabled for delegation, the **gss_init_sec_context()** request will be successful but the **GSS_C_DELEG_FLAG** will not be set in the returned flags. You can use the **klist** command with the **-f** option to display the ticket flags. The ticket-granting ticket must have the **F** flag set and the service ticket must have the **o** flag set.

API introduced: V5R1

---

gss_inquire_context()—Get Information About Security Context

Syntax

```
#include <gssapi.h>

OM_uint32 gss_inquire_context (
    OM_uint32 *    minor_status,
    gss_ctx_id_t     context_handle,
    gss_name_t *   source_name,
    gss_name_t *     target_name,
    OM_uint32 *    lifetime,
    gss_OID *      mech_type,
    gss_flags_t *    ret_flags,
    int *      local,
    int *      open);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_inquire_context()** function returns information about a security context to the calling application.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The handle for the security context.

source_name (Output)
The principal name associated with the context initiator. Specify **NULL** for this parameter if the principal name is not required.

target_name (Output)
The principal name associated with the context acceptor. Specify **NULL** for this parameter if the principal name is not required.

lifetime (Output)
The number of seconds for which the context remains valid. Specify **NULL** for this parameter if the context lifetime is not required. The returned value is **GSS_C_INDEFINITE** if the security mechanism does not support context expiration.

mech_type (Output)
The mechanism used to create the security context. The gss_OID value returned for this parameter points to read-only storage and must not be released by the application. Specify **NULL** for this parameter if the mechanism type is not required.

ret_flags (Output)
A bit mask containing independent flags indicating which GSS services are available for the context. Specify **NULL** for this parameter if the available service flags are not required. The following symbolic definitions are provided to test the individual flags and should be logically ANDed with the value of *ret_flags* to test whether the context supports the service options:

GSS_C_ANON_FLAG	The initiator identity will not be provided to the context acceptor.
GSS_C_CONF_FLAG	Message confidentiality services are available.
GSS_C_DELEG_FLAG	Delegated credentials will be available to the context acceptor.
GSS_C_INTEG_FLAG	Message integrity services are available.
GSS_C_MUTUAL_FLAG	Mutual authentication will be performed. The **gss_accept_sec_context()** routine will generate an output token which the context acceptor must return to the context initiator to complete the security context setup.
GSS_C_PROT_READY_FLAG	Protection services, as specified by the states of the **GSS_C_CONF_FLAG** and **GSS_C_INTEG_FLAG**, are available for use even if the context is not fully established. Otherwise, protection services are available for use only if value returned by the *open* parameter is TRUE.
GSS_C_REPLAY_FLAG	Message replay detection will be performed.
GSS_C_SEQUENCE_FLAG	Message sequence checking will be performed.

local (Output)
TRUE if the context was initiated locally and **FALSE** otherwise. Specify **NULL** for this parameter if the local indication is not required.

open (Output)
TRUE if context establishment has been completed and **FALSE** otherwise. Specify **NULL** for this parameter if the open indication is not required.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The referenced context has expired.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context handle provided by the caller does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

---

gss_inquire_cred()—Get Information About GSS Credential

Syntax

```
#include <gssapi.h>

OM_uint32 gss_inquire_cred (
    OM_uint32 *    minor_status,
    gss_cred_id_t      cred_handle,
    gss_name_t *   name,
    OM_uint32 *    lifetime,
    gss_cred_usage_t *   cred_usage,
    gss_OID_set *    mechanisms);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_inquire_cred()** function returns information about a GSS credential to the calling application. If **GSS_C_NO_CREDENTIAL** is specified for the *cred_handle* parameter, the default security mechanism is used to process the request.

Parameters

minor_status (Output)
A status code from the security mechanism.

cred_handle (Input)
The handle for the GSS credential. Specify **GSS_C_NO_CREDENTIAL** to get information about the default credential for the default security mechanism.

name (Output)
The principal name associated with the credential. Specify **NULL** for this parameter if the principal name is not required.

lifetime (Output)
The number of seconds for which the credential remains valid. The returned value is zero if the credential has expired. Specify **NULL** for this parameter if the credential lifetime is not required.

cred_usage (Output)
One of the following values describing how the application can use the credential. Specify **NULL** for this parameter if the credential usage is not required.

GSS_C_INITIATE	The application may initiate a security context.
GSS_C_ACCEPT	The application may accept a security context.
GSS_C_BOTH	The application may both initiate and accept security contexts.

mechanisms (Output)
The set of security mechanisms supported by the credential. Specify **NULL** for this parameter if

the mechanism set is not required. The gss_OID_set returned for this parameter should be released when it is no longer needed by calling the **gss_release_oid_set()** routine.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CREDENTIALS_EXPIRED
The credentials have expired. Credential information is still returned for an expired credential, but the *lifetime* value is returned as zero.

GSS_S_DEFECTIVE_CREDENTIAL
The credentials are not valid.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The *cred_handle* parameter does not refer to a valid credential or there are no default credentials available.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R
Each directory in the path name preceding the credential cache file	*X
Credential cache file	*RW

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_inquire_cred_by_mech()—Get Information About GSS Credential for Single Security Mechanism

Syntax

```
#include <gssapi.h>

OM_uint32 gss_inquire_cred_by_mech (
    OM_uint32 *    minor_status,
    gss_cred_id_t      cred_handle,
    gss_OID      mech_type,
```

```
gss_name_t *   name,
OM_uint32 *    init_lifetime,
OM_uint32 *    accept_lifetime,
gss_cred_usage_t *   cred_usage);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_inquire_cred_by_mech()** function returns information about a GSS credential for a single security mechanism. The information is obtained using the specified security mechanism.

Parameters

minor_status (Output)
A status code from the security mechanism.

cred_handle (Input)
The handle for the GSS credential. Specify **GSS_C_NO_CREDENTIAL** to get information about the default credential for the default security mechanism.

mech_type (Input)
The mechanism to be used to obtain the returned information as follows:

gss_mech_krb5_old Beta Kerberos V5 mechanism
gss_mech_krb5 Kerberos V5 mechanism

name (Output)
The principal name associated with the credential. Specify **NULL** for this parameter if the principal name is not required.

init_lifetime (Output)
The number of seconds for which the credential remains valid for initiating contexts. Specify **NULL** for this parameter if the credential lifetime is not required.

accept_lifetime (Output)
The number of seconds for which the credential remains valid for accepting contexts. Specify **NULL** for this parameter if the credential lifetime is not required.

cred_usage (Output)
One of the following values describing how the application can use the credential. Specify **NULL** for this parameter if the credential usage is not required.

GSS_C_ACCEPT The application may accept a security context.
GSS_C_BOTH The application may both initiate and accept security contexts.
GSS_C_INITIATE The application may initiate a security context.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_MECH
The requested mechanism is not supported.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CREDENTIALS_EXPIRED
The credentials have expired. Credential information will still be returned for an expired credential, but the *lifetime* value will be returned as zero.

GSS_S_DEFECTIVE_CREDENTIAL
The credentials are not valid.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The *cred_handle* parameter does not refer to a valid credential or there are no default credentials available.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_inquire_mechs_for_name()—Determine Mechanisms to Process Name

Syntax

```
#include <gssapi.h>

OM_uint32 gss_inquire_mechs_for_name (
    OM_uint32 *    minor_status,
    gss_name_t     input_name,
    gss_OID_set *    mech_types);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_inquire_mechs_for_name()** function returns the mechanisms with which a name may be processed.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_name (Input)
The name to be queried.

mech_types (Output)
The mechanisms that can be used with the specified name. The gss_OID_set returned for this parameter should be released by calling the **gss_release_oid_set()** routine when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_NAME
The supplied name is not valid.

GSS_S_BAD_NAMETYPE
The name type is not supported.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_inquire_names_for_mech()—Get Name Types Supported by Security Mechanism

Syntax

```
#include <gssapi.h>

OM_uint32 gss_inquire_names_for_mech (
    OM_uint32 *    minor_status,
    gss_OID      mech_type,
    gss_OID_set *    mech_names);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_inquire_names_for_mech()** function returns the name types supported by a security mechanism.

Parameters

minor_status (Output)
A status code from the security mechanism.

mech_type (Input)
The mechanism to be queried as follows:

gss_mech_krb5_old Beta Kerberos V5 mechanism
gss_mech_krb5 Kerberos V5 mechanism

mech_names (Output)
The name types supported by the specified mechanism. The gss_OID_set returned for this parameter should be released by calling the **gss_release_oid_set()** routine when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_MECH
The requested mechanism is not supported.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_krb5_get_ccache()—Get Kerberos Protocol Credentials Cache Associated with Specified GSS Credential

Syntax

```
#include <gssapi.h>

OM_uint32 gss_krb5_get_ccache (
    OM_uint32 *    minor_status,
    gss_cred_id_t      cred_handle,
    krb5_ccache *    ccache);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_krb5_get_ccache()** function returns the returns the handle for the Kerberos credentials cache associated with a GSS credential. The application must not close or destroy this credentials cache. The returned handle is longer be valid once the GSS credential has been released.

Parameters

minor_status (Output)
A status code from the security mechanism.

cred_handle (Input)
The handle for the GSS credential.

ccache (Output)
The handle for the credentials cache. A NULL value is returned if there is no credentials cache associated with the GSS credential.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The credential handle does not refer to a valid GSS credential.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_krb5_acquire_cred_cache()—Acquire GSS Credential from a Kerberos Protocol Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code gss_krb5_acquire_cred_cache (
   OM_uint32 *                          minor_status,
   krb5_ccache                          ccache,
   OM_uint32                            time_req,
   gss_cred_usage_t                     cred_usage,
   gss_cred_id_t *                      output_cred_handle,
   OM_uint32 *                          time_rec)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_krb5_acquire_cred_ccache()** routine acquires a GSS-API credential using a Kerberos credentials cache. This function allows an application to obtain a GSS-API credential for use with the Kerberos mechanism. The application can then use the credential with the **gss_init_sec_context()** and **gss_accept_sec_context()** routines. The Kerberos credentials cache must not be closed until the GSS-API credential is no longer needed and has been deleted.

If GSS_C_INITIATE or GSS_C_BOTH is specified for the credential usage, the application must have a valid ticket in the credentials cache and the ticket must not expire for at least 10 minutes. The **gss_krb5_acquire_cred_ccache()** routine will use the first valid ticket-granting ticket (or the first valid service ticket if there is no TGT) to create the GSS-API credential.

If GSS_C_ACCEPT or GSS_C_BOTH is specified for the credential usage, the principal associated with the GSS-API credential must be defined in a key table. The KRB5_KTNAME environment variable is used to identify the key table used by the Kerberos security mechanism.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

ccache (Input)
The Kerberos credentials cache to be used for the credential. The principal name for the GSS-API credential is obtained from the credentials cache. The credentials cache must contain a valid ticket-granting ticket for this principal if a GSS_C_INITIATE or GSS_C_BOTH credential is Requested.

time_req (Input)
The number of seconds that the credential remains valid. Specify GSS_C_INDEFINITE to request the maximum credential lifetime. Specify zero for the default lifetime of 2 hours. The actual credential lifetime will be limited by the lifetime of the underlying ticket-granting ticket for GSS_C_INITIATE and GSS_C_BOTH credentials.

cred_usage (Input)
The desired credential usage as follows:

- GSS_C_INITIATE if the credential can be used only to initiate security contexts.
- GSS_C_ACCEPT if the credential can be used only to accept security contexts.
- GSS_C_BOTH if the credential can be used to both initiate and accept security contexts.

output_cred_handle (Output)
The handle returned for the GSS-API credential.

time_rec (Output)
The number of seconds returned for which the credential will remain valid. If the time remaining is not required, specify NULL for this parameter.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_BAD_MECH
None of the requested mechanisms are supported by the local system.

GSS_S_NO_CRED
The Kerberos credentials cache does not contain a valid ticket-granting ticket.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_krb5_ccache_name()—Set Default Kerberos Protocol Credentials Cache Name

Syntax

```
#include <krb5.h>

krb5_error_code gss_krb5_ccache_name (
   OM_uint32 *                          minor_status,
   char *                               new_name,
   char **                              old_name)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_krb5_ccache_name()** routine sets the default credentials cache name for use by the Kerberos mechanism. The default credentials cache is used by **gss_acquire_cred()** to create a GSS-API credential. It is also used by **gss_init_sec_context()** when GSS_C_NO_CREDENTIAL is specified for the GSS-API credential used to establish the security context.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

new_name (Input)
The new name for the default GSS-API Kerberos credentials cache.

old_name (Output)
The name returned of the current default credentials cache or NULL if the default credentials cache has not been set. Specify NULL for this parameter if you do not need the current credentials cache name. The returned name should be released by calling **krb5_free_string()** when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_krb5_copy_ccache()—Copy Tickets From Associated GSS Credentials to Kerberos Protocol Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code gss_krb5_copy_ccache (
   OM_uint32 *                          minor_status,
   gss_cred_id_t                        cred_handle,
   krb5_ccache                          ccache)
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_krb5_copy_ccache()** routine will copy the tickets from the Kerberos credentials cache associated with a GSS-API credential to a credentials cache provided by the caller. The supplied Kerberos credentials cache must have been initialized by **krb5_cc_initialize()** before calling **gss_krb5_copy_ccache()**. The GSS-API credential must have been created by specifying GSS_C_INITIATE or GSS_C_BOTH.

Parameters

minor_status (Output)
Status code returned from the security mechanism.

cred_handle (Input)
The GSS-API credential handle. This must be a GSS_C_INITIATE or GSS_C_BOTH credential.

ccache (Input)
The Kerberos credentials cache.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons which are not defined at the GSS level. The minor_status return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The credential handle does not refer to a valid GSS-API credential.

Authorities

None.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA081 E	Unable to set return value or error code.

API introduced: V5R2

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_krb5_get_tkt_flags()—Get Kerberos Protocol Ticket Flags

Syntax

```
#include <gssapi.h>

OM_uint32 gss_krb5_get_tkt_flags (
    OM_uint32 *    minor_status,
    gss_ctx_id_t     context_handle,
    krb5_flags *   tkt_flags);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_krb5_get_tkt_flags()** function returns the Kerberos ticket flags from the Kerberos ticket associated with the security context. Refer to the Kerberos protocol-specific API documentation for a description of the various flags.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The handle for the GSS security context.

tkt_flags (Output)
The ticket flags from the Kerberos ticket associated with the security context.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context handle does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_oid_to_str()—Convert OID Object to String Representation of Object

Syntax

```
#include <gssapi.h>

OM_uint32 gss_oid_to_str(
    OM_uint32 *    minor_status,
    gss_OID      input_oid,
    gss_buffer_t   output_string);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_oid_to_str()** function converts a gss_oid object to a string representation of the object identifier. The string representation consists of a series of blank-separated numbers enclosed in braces. The **gss_str_to_oid()** routine can be used to convert the string representation back to a gss_oid object.

Not every coded character set identifier (CCSID) contains the left and right brace characters; however, alternative CCSID values often are available. For example, instead of using Greece 423, run the job with a default CCSID of 875.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_oid (Input)
The gss_OID to be converted.

output_string (Output)
The string representation of the object identifier. The gss_buffer_t returned for this parameter should be released by calling the **gss_release_buffer()** routine when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

gss_process_context_token()—Process Received Context Token

Syntax

```
#include <gssapi.h>

OM_uint32 gss_process_context_token(
    OM_uint32 *    minor_status,
    gss_ctx_id_t   context_handle
    gss_buffer_t   input_token);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_process_context_token()** function processes a context token received from the partner application.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The context that should be used when processing the token.

input_token (Input)
The token received from the partner application.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_SIG
The token signature was not correct.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_DEFECTIVE_TOKEN
Consistency checks performed on the input token failed.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context handle does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. Tokens are usually associated with either the context establishment or with message security services. If the tokens are associated with the context establishment, they are processed by the **gss_init_sec_context()** and **gss_accept_sec_context()** routines. If the tokens are associated with message security services, they are processed by the **gss_verify_mic()** and **gss_unwrap()** routines. Tokens generated by the **gss_delete_sec_context()** routine, however, are processed by the **gss_process_context_token()** routine.

API introduced: V5R1

---

gss_release_buffer()—Release Storage Associated with Buffer

Syntax

```
#include <gssapi.h>

OM_uint32 gss_release_buffer(
    OM_uint32 *    minor_status,
    gss_buffer_t   buffer);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_release_buffer()** function releases storage associated with a gss_buffer_t buffer. The gss_buffer_desc structure itself is not released.

Parameters

minor_status (Output)
A status code from the security mechanism.

buffer (Input/Output)
The buffer to be released. Upon successful completion, the *length* and *value* fields are set to zero.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_release_cred()—Release Storage Associated with GSS Credential

Syntax

```
#include <gssapi.h>

OM_uint32 gss_release_cred(
    OM_uint32 *    minor_status,
    gss_cred_id_t *    cred_handle);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_release_cred()** function releases the local data structures associated with a GSS credential. If **gss_c_no_credential** is specified for the *cred_handle* parameter, no credential is released and **gss_s_complete** is returned for the major status return value.

Parameters

minor_status (Output)
A status code from the security mechanism.

cred_handle (Input/Output)
The credential to be released. Upon successful completion, the *cred_handle* value is set to **GSS_C_NO_CREDENTIAL**. If the *cred_handle* value is **GSS_C_NO_CREDENTIAL**, the major status return value is **GSS_S_COMPLETE** and nothing is released.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_DEFECTIVE_CREDENTIAL
Consistency checks performed on the credential structure failed.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CRED
The *cred_handle* parameter does not refer to a valid credential.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_release_name()—Release Storage Associated with GSS Internal Name

Syntax

```
#include <gssapi.h>

OM_uint32 gss_release_name(
    OM_uint32 *    minor_status,
    gss_name_t *   name);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_release_name()** function releases storage associated with a gss_name_t internal name.

Parameters

minor_status (Output)
A status code from the security mechanism.

name (Input/Output)
The name to be released. Upon successful completion, the *name* value is set to **GSS_C_NO_NAME**.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_NAME
The specified name is not valid.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_release_oid()—Release Storage Associated with OID Object

Syntax

```
#include <gssapi.h>

OM_uint32 gss_release_oid(
    OM_uint32 *    minor_status,
    gss_OID *      oid);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_release_oid()** function releases storage associated with a gss_oid object. The **gss_release_oid()** routine must not be called to release a read-only oid that was returned by one of the GSS routines.

Parameters

minor_status (Output)
A status code from the security mechanism.

oid (Input/Output)
The gss_OID to be released. Upon successful completion, the *oid* value is set to **GSS_C_NO_OID**.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_release_oid_set()—Release Storage Associated with a Set of OID Objects

Syntax

```
#include <gssapi.h>

OM_uint32 gss_release_oid_set(
    OM_uint32 *    minor_status,
    gss_OID_set *    oid_set);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_release_oid_set()** function releases storage associated with a gss_oid_set object.

Parameters

minor_status (Output)
A status code from the security mechanism.

oid (Input/Output)
The gss_OID_set to be released. Upon successful completion, the *oid_set* value is set to **GSS_C_NO_OID_SET**.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_str_to_oid()—Convert String Representation of an Object Identifier to an Internal OID Object

Syntax

```
#include <gssapi.h>

OM_uint32 gss_str_to_oid(
    OM_uint32 *    minor_status,
    gss_buffer_t   input_string,
    gss_OID *      output_oid);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_str_to_oid()** function converts the string representation of an object identifier to a gss_OID object.

Parameters

minor_status (Output)
A status code from the security mechanism.

input_string (Input)
The string to be converted.

output_oid (Output)
The object identifier. The gss_OID returned for this parameter should be released by calling the **gss_release_oid()** routine when it is no longer needed.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. A string representation is a series of blank-separated or period-separated numbers enclosed in braces. For example, the Kerberos V5 security mechanism object identifier is represented as {1 2 840 113554 1 2 2}.

   Not every coded character set identifier (CCSID) contains the left and right brace characters; however, alternative CCSID values often are available. For example, instead of using Greece 423, run the job with a default CCSID of 875.
2. While the blank-separated form should be used for portability, the **gss_str_to_oid()** routine also accepts the period-separated form for compatibility with other applications. The **gss_oid_to_str()** routine, however, always generates the blank-separated form.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_test_oid_set_member()—Determine if Specified OID is Contained in a Specified OID Set

Syntax

```
#include <gssapi.h>

OM_uint32 gss_test_oid_set_member(
    OM_uint32 *    minor_status,
    gss_OID      member_oid,
    gss_OID_set    oid_set,
    int *      is_present);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_test_oid_set_member()** function checks an oid set to see if a specified oid is a member of the set.

Parameters

minor_status (Output)
A status code from the security mechanism.

member_oid (Input)
The OID to search for in the OID set.

oid_set (Input)
The OID set to check.

is_present (Output)
1 if the OID is a member of the OID set. Otherwise, it is set to zero.

Return Value

The return value is one of the following status codes:

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. The**gss_create_empty_oid_set()** routine can be used to create an empty oid set. The **gss_add_oid_set_member()** routine can be used to add an oid to an existing oid set.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_unwrap()—Unwrap a Message

Syntax

```
#include <gssapi.h>

OM_uint32 gss_unwrap (
    OM_uint32 *    minor_status,
    gss_ctx_id_t     context_handle,
    gss_buffer_t   input_message,
    gss_buffer_t   output_message,
    int *      conf_state,
    gss_qop_t *    qop_state);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_unwrap()** function unwraps a message sealed by the **gss_wrap()** routine and verifies the embedded signature. The *conf_state* return parameter indicates whether or not the message has been encrypted.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The context in which the message arrived.

input_message (Input)
The sealed message token generated by the **gss_wrap()** routine.

output_message (Output)
The unsealed message.

conf_state (Output)
The level of confidentiality that was applied to the message. Specify **NULL** for this parameter if the confidentiality state is not needed. The return value is set as follows:

TRUE	Both confidentiality and integrity services were applied.
FALSE	Only integrity services were applied.

qop_state (Output)
The quality of protection that was applied to the message. Specify **NULL** for this parameter if the quality of protection is not needed.

The Kerberos security mechanism supports three quality of protection levels as follows (in decreasing order or speed):

GSS_KRB5_INTEG_C_QOP_MD5	Truncated MD5
GSS_KRB5_INTEG_C_QOP_DES_MD5	DES_MAC of an MD5 hash
GSS_KRB5_INTEG_C_QOP_DES_MAC	Normal DES_MAC algorithm

Return Value

The return value is one of the following status codes:

GSS_S_BAD_SIG
The input token contains an incorrect signature.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The context identifier provided by the caller has expired.

GSS_S_CREDENTIALS_EXPIRED
Credentials are no longer valid.

GSS_S_DEFECTIVE_TOKEN
Consistency checks performed on the input token failed.

GSS_S_DUPLICATE_TOKEN
The token is a duplicate of a token that has already been processed.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_GAP_TOKEN
One or more predecessor tokens have not been processed.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

GSS_S_OLD_TOKEN
The token is too old to be checked for duplication against previous tokens. This is a fatal error during context establishment.

GSS_S_UNSEQ_TOKEN
A later token has already been processed.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_verify_mic()—Verify that Cryptographic Signature is Correct

Syntax

```
#include <gssapi.h>

OM_uint32 gss_verify_mic (
    OM_uint32 *  minor_status,
    gss_ctx_id_t   context_handle,
    gss_buffer_t  input_message,
    gss_buffer_t  input_token,
    gss_qop_t *  qop_state);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_verify_mic()** function verifies that the cryptographic signature for a message is correct. This ensures that the message has not been modified since the signature was generated.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The context in which the message arrived.

input_message (Input)
The message to be verified.

input_token (Input)
The signature token generated by the **gss_get_mic()** routine.

qop_state (Output)
The quality of protection that was applied to the message. Specify **NULL** for this parameter if the quality of protection is not needed.

The Kerberos security mechanism supports three quality of protection levels as follows:

GSS_KRB5_INTEG_C_QOP_MD5	Truncated MD5
GSS_KRB5_INTEG_C_QOP_DES_MD5	DES_MAC of an MD5 hash
GSS_KRB5_INTEG_C_QOP_DES_MAC	Normal DES_MAC algorithm

Return Value

The return value is one of the following status codes:

GSS_S_BAD_SIG
The input token contains an incorrect signature.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The context identifier provided by the caller has expired.

GSS_S_CREDENTIALS_EXPIRED
The credentials associated with the referenced context have expired.

GSS_S_DEFECTIVE_CREDENTIAL
The credential is defective.

GSS_S_DEFECTIVE_TOKEN
Consistency checks performed on the input token failed.

GSS_S_DUPLICATE_TOKEN
The token is a duplicate of a token that has already been processed.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_GAP_TOKEN
One or more predecessor tokens have not been processed.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

GSS_S_OLD_TOKEN
The token is too old to be checked for duplication against previous tokens. This is a fatal error during context establishment.

GSS_S_UNSEQ_TOKEN
A later token has already been processed.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

---

gss_wrap()—Cryptographically Sign and Optionally Encrypt Message

Syntax

```
#include <gssapi.h>

OM_uint32 gss_wrap (
    OM_uint32 *    minor_status,
    gss_ctx_id_t     context_handle,
    int      conf_req,
    gss_qop_t      qop_req,
    gss_buffer_t   input_message,
    int *      conf_state,
    gss_buffer_t   output_message);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_wrap()** function cryptographically signs and optionally encrypts a message. The token returned in the *output_message* parameter contains both the signature and the message. This token is then sent to the partner application that calls the **gss_unwrap()** routine to extract the original message and verify its authenticity.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The context handle to be associated with the message when it is sent to the partner application.

conf_req (Input)
The requested level of confidentiality and integrity services as follows:

TRUE Both confidentiality and integrity services are requested.

FALSE Only integrity services are requested.

qop_req (Input)
The requested quality of protection for the message. Specify **GSS_C_QOP_DEFAULT** to use the default quality of protection as defined by the selected security mechanism.

The Kerberos security mechanism supports three quality of protection levels as follows (in decreasing order or speed). Specify **GSS_KRB5_INTEG_C_QOP_DES_MD5** (or **GSS_C_QOP_DEFAULT**) for interoperability with other implementations of the Kerberos security mechanism.

GSS_KRB5_INTEG_C_QOP_MD5	Truncated MD5
GSS_KRB5_INTEG_C_QOP_DES_MD5	DES_MAC of an MD5 hash (default)
GSS_KRB5_INTEG_C_QOP_DES_MAC	Normal DES_MAC algorithm

input_message (Input)
The message to be wrapped.

conf_state (Output)
The level of confidentiality that was applied to the message. Specify **NULL** for this parameter if the confidentiality state is not needed. The return value is set as follows:

TRUE	Both confidentiality and integrity services were applied.
FALSE	Only integrity services were applied.

output_message (Output)
The wrapped message. The buffer should be released when it is no longer needed by calling the **gss_release_buffer()** routine.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_QOP
The quality of protection value is not valid.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The context identifier provided by the caller has expired.

GSS_S_CREDENTIALS_EXPIRED
Credentials are no longer valid.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X

Object Referred to	Data Authority Required
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Usage Notes

1. If confidentiality is requested (the *conf_req* is **true**) but confidentiality services are not available for the security context, no error is returned and only integrity services are performed. The *conf_state* return parameter indicates whether or not the requested confidentiality services were performed.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

---

gss_wrap_size_limit()—Determine Largest Message that can be Wrapped

Syntax

```
#include <gssapi.h>

OM_uint32 gss_wrap_size_limit (
    OM_uint32 *    minor_status,
    gss_ctx_id_t     context_handle,
    int      conf_req,
    gss_qop_t      qop_req,
    OM_uint32      size_req,
    OM_uint32 *    max_size);
```

Service Program Name: QSYS/QKRBGSS
Default public authority: *USE
Threadsafe: Yes

The **gss_wrap_size_limit()** function determines the largest message that can be processed by the **gss_wrap()** routine without exceeding the specified output token size.

Parameters

minor_status (Output)
A status code from the security mechanism.

context_handle (Input)
The security context that will be associated with the messages.

conf_req (Input)
Whether confidentiality services will be requested for the messages as follows:

TRUE	Both confidentiality and integrity and authentication services will be requested.
FALSE	Only integrity and authentication services will be requested.

qop_req (Input)
The quality of protection that will be used with the messages. Specify **GSS_C_QOP_DEFAULT** to use the default quality of protection as defined by the selected security mechanism.

The Kerberos security mechanism supports three quality of protection levels as follows (in decreasing order or speed):

GSS_KRB5_INTEG_C_QOP_MD5	Truncated MD5
GSS_KRB5_INTEG_C_QOP_DES_MD5	DES_MAC of an MD5 hash (default)
GSS_KRB5_INTEG_C_QOP_DES_MAC	Normal DES_MAC algorithm

size_req (Input)
The maximum output token size.

max_size (Output)
The maximum message size that can be processed without exceeding the specified maximum token size.

Return Value

The return value is one of the following status codes:

GSS_S_BAD_QOP
The quality of protection requested is not valid.

GSS_S_COMPLETE
The routine completed successfully.

GSS_S_CONTEXT_EXPIRED
The context identifier provided by the caller has expired.

GSS_S_FAILURE
The routine failed for reasons that are not defined at the GSS level. The *minor_status* return parameter contains a mechanism-dependent error code describing the reason for the failure.

GSS_S_NO_CONTEXT
The context identifier provided by the caller does not refer to a valid security context.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the configuration file	*X
Configuration file	*R

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

API introduced: V5R1

Top | Security APIs | UNIX-Type APIs | APIs by category

qkrb_build_spnego_init_token()—Build a SPNEGO Initiator Token

Syntax

```
#include <qkrbspnego.h>

OM_uint32 qkrb_build_spnego_init_token(
    gss_OID_set          supported_mechanisms,
    gss_flags_t        * context_flags,
    gss_buffer_desc    * token_for_first_mechanism,
    gss_buffer_desc    * mechanism_list_mic,
    OM_uint32            format_type,
    gss_buffer_desc    * initiator_token);
```

Service Program Name: QSYS/QKRBSPNEGO
Default Public Authority: *USE
Threadsafe: Yes

The **qkrb_build_spnego_init_token()** builds a Simple and Protected GSS-API Negotiation (SPNEGO) Initiator Token and returns the results to the caller.

Authorities

No authorities are required.

Parameters

supported_mechanisms (Input)
A gss_OID_set that contains one or more security mechanisms supported by the initiator. Specify GSS_C_NO_OID_SET if there are no mechanisms to add.

context_flags (Input)
The context flags that are required to establish the context. The context flags should be filled in from the req_flags parameter of gss_init_sec_context(). Specify *NULL* for this parameter if there are no context flags to send.

The following flags are supported. All other flags will be ignored.

GSS_C_ANON_FLAG (64)	The initiator identity will not be provided to the context acceptor.
GSS_C_CONF_FLAG (16)	Message confidentiality services are available.
GSS_C_DELEG_FLAG (1)	Delegated credentials will be available to the context acceptor.
GSS_C_INTEG_FLAG (32)	Message integrity services are available.
GSS_C_MUTUAL_FLAG (2)	Mutual authentication will be performed. The **gss_accept_sec_context()** routine will generate an output token which the context acceptor must return to the context initiator to complete the security context setup.
GSS_C_REPLAY_FLAG (4)	Message replay detection will be performed.
GSS_C_SEQUENCE_FLAG (8)	Message sequence checking will be performed.

token_for_first_mechanism (Input)
The security token associated with the first mechanism in the supported_mechanisms gss_OID_set. Specify GSS_C_NO_BUFFER if there is no token.

mechanism_list_mic (Input)
The mechanism list MIC to be added to the initiator token. Specify GSS_C_NO_BUFFER if there is no mechanism list MIC.

format_type (Input)
The format to follow when building the SPNEGO token. Possible values are:

GSS_SPNEGO_FORMAT_0 (0) The format of the SPNEGO token built follows the syntax defined in RFC 2478.
GSS_SPNEGO_FORMAT_1 (1) The format of the SPNEGO token built follows the syntax defined in RFC 2478 with one exception. The mechanism_list_mic is sent as SEQUENCE/ GENERAL_STRING.

initiator_token (Output)
The initiator token built from the input information. The application should release the initiator token when it is no longer needed by calling the **gss_release_buffer()** routine.

Return Value

The return value is one of the following status codes:

GSS_SPNEGO_SUCCESS (0)
The routine completed successfully.

GSS_SPNEGO_UNEXPECTED_ERR (1)
The routine failed for unexpected reasons. Check the joblog for errors.

GSS_SPNEGO_NOMEM (2)
Memory allocation failed.

Related Information

For a description of the SPNEGO protocol, see RFC 2478 on the RFC Pages for *The Simple and Protected GSS-API Negotiation Mechanism*.

API introduced: V5R4

Top | Security APIs | UNIX-Type APIs | APIs by category

---

qkrb_build_spnego_target_token()—Build a SPNEGO Target Token

Syntax

```
#include <qkrbspnego.h>

OM_uint32 qkrb_build_spnego_target_token(
    gss_buffer_desc    * negotiation_result,
    gss_OID_desc       * supported_mechanism,
    gss_buffer_desc    * response_token,
    gss_buffer_desc    * mechanism_list_mic,
    OM_uint32            format_type,
    gss_buffer_desc    * target_response_token);
```

Service Program Name: QSYS/QKRBSPNEGO
Default Public Authority: *USE
Threadsafe: Yes

The **qkrb_build_spnego_target_token()** builds a Simple and Protected GSS-API Negotiation (SPNEGO) Target Token and returns the results to the caller.

Authorities

No authorities are required.

Parameters

negotiation_result (Input)
Result of the SPNEGO negotiation exchange, specified by the target. Possible values are:

GSS_SPNEGO_ACCEPT_COMPLETED (0x00)	The target accepts the preferred security mechanism, and the context is established for the mechanism.
GSS_SPNEGO_ACCEPT_INCOMPLETE (0x01)	The target accepts one of the proposed security mechanisms and further exchanges are necessary.
GSS_SPNEGO_REJECTED (0x02)	The target rejects all the proposed security mechanisms.

supported_mechanism (Input)
The supported mechanism to be added to the target token. Specify GSS_C_NO_BUFFER if there is no supported mechanism.

response_token (Input)
The response token to be added to the target token. Specify GSS_C_NO_BUFFER if there is no response token.

mechanism_list_mic (Input)
The mechanism list MIC to be added to the target token. Specify GSS_C_NO_BUFFER if there is no mechanism list MIC.

format_type (Input)
The format to follow when building the SPNEGO token. Possible values are:

GSS_SPNEGO_FORMAT_0 (0)	The format of the SPNEGO token built follows the syntax defined in RFC 2478.
GSS_SPNEGO_FORMAT_1 (1)	The format of the SPNEGO token built follows the syntax defined in RFC 2478 with one exception. The mechanism_list_mic is sent as SEQUENCE/ GENERAL_STRING.

target_response_token (Output)
The target token built from the input information. The application should release the target token when it is no longer needed by calling the **gss_release_buffer()** routine.

Return Value

The return value is one of the following status codes:

GSS_SPNEGO_SUCCESS (0)
The routine completed successfully.

GSS_SPNEGO_UNEXPECTED_ERR (1)
The routine failed for unexpected reasons. Check the joblog for errors.

GSS_SPNEGO_NOMEM (2)
Memory allocation failed.

Related Information

For a description of the SPNEGO protocol, see RFC 2478 on the RFC Pages for *The Simple and Protected GSS-API Negotiation Mechanism*.

API introduced: V5R4

Top | Security APIs | UNIX-Type APIs | APIs by category

qkrb_parse_spnego_init_token()—Parse a SPNEGO Initiator Token

Syntax

```
#include <qkrbspnego.h>

OM_uint32 qkrb_parse_spnego_init_token(
    gss_buffer_desc     * init_token,
    qkrb_spnego_init_item_t **     parsed_token);
```

Service Program Name: QSYS/QKRBSPNEGO
Default Public Authority: *USE
Threadsafe: Yes

The **qkrb_parse_spnego_init_token()** function parses a Simple and Protected GSS-API Negotiation (SPNEGO) Initiator Token and returns the results to the caller.

Authorities

No authorities are required.

Parameters

init_token (Input)
The SPNEGO initiator token encoded in Abstract Syntax Notation 1 Distinguished Encoding Rules (ASN.1 DER) format.

parsed_token (Output)
The information parsed from the initiator token. The application should release the parsed token information when it is no longer needed by calling the **qkrb_free_spnego_init_item()** routine.

Return Value

The return value is one of the following status codes:

GSS_SPNEGO_SUCCESS (0)
The routine completed successfully.

GSS_SPNEGO_UNEXPECTED_ERR (1)
The routine failed for unexpected reasons. Check the joblog for errors.

GSS_SPNEGO_NOMEM (2)
Memory allocation failed.

GSS_SPNEGO_DEFECTIVE_TOKEN (3)
Consistency checks performed on the input token failed.

GSS_SPNEGO_INCOMPLETE_RESULTS (4)
Unable to successfully parse all token items. Partial results have been returned.

Related Information

For a description of the SPNEGO protocol, see RFC 2478 on the RFC Pages for *The Simple and Protected GSS-API Negotiation Mechanism*.

API introduced: V5R4

Top | Security APIs | UNIX-Type APIs | APIs by category

---

qkrb_parse_spnego_target_token()—Parse a SPNEGO Target Token

Syntax

```
#include <qkrbspnego.h>

OM_uint32 qkrb_parse_spnego_target_token(
    gss_buffer_desc     * target_token,
    qkrb_spnego_target_item_t **     parsed_token);
```

Service Program Name: QSYS/QKRBSPNEGO
Default Public Authority: *USE
Threadsafe: Yes

The **qkrb_parse_spnego_target_token()** function parses a Simple and Protected GSS-API Negotiation (SPNEGO) Target Token and returns the results to the caller.

Authorities

No authorities are required.

Parameters

target_token (Input)
The SPNEGO target token encoded in Abstract Syntax Notation 1 Distinguished Encoding Rules (ASN.1 DER) format.

parsed_token (Output)
The information parsed from the target token. The application should release the parsed token information when it is no longer needed by calling the **qkrb_free_spnego_target_item()** routine.

Return Value

The return value is one of the following status codes:

GSS_SPNEGO_SUCCESS (0)
The routine completed successfully.

GSS_SPNEGO_UNEXPECTED_ERR (1)
The routine failed for unexpected reasons. Check the joblog for errors.

GSS_SPNEGO_NOMEM (2)
Memory allocation failed.

GSS_SPNEGO_DEFECTIVE_TOKEN (3)
Consistency checks performed on the input token failed.

GSS_SPNEGO_INCOMPLETE_RESULTS (4)
Unable to successfully parse all token items. Partial results have been returned.

Related Information

For a description of the SPNEGO protocol, see RFC 2478 on the RFC Pages for *The Simple and Protected GSS-API Negotiation Mechanism*.

API introduced: V5R4

Top | Security APIs | UNIX-Type APIs | APIs by category

---

qkrb_free_spnego_init_item()—Release Storage Associated with an Initiator Token Item

Syntax

```
#include <qkrbspnego.h>

OM_uint32 qkrb_free_spnego_init_item(
    qkrb_spnego_init_item_t **    parsed_token);
```

Service Program Name: QSYS/QKRBSPNEGO
Default Public Authority: *USE
Threadsafe: Yes

The **qkrb_free_spnego_init_item()** function releases storage associated with a qkrb_spnego_init_item_t object.

Authorities

No authorities are required.

Parameters

parsed_token (Input/Output)
The qkrb_spnego_init_item_t to be released. Upon successful completion, the *parsed_token* value is set to NULL.

Return Value

The return value is one of the following status codes:

GSS_SPNEGO_SUCCESS (0)
The routine completed successfully.

GSS_SPNEGO_UNEXPECTED_ERR (1)
The routine failed for unexpected reasons. Check the joblog for errors.

API introduced: V5R4

Top | Security APIs | UNIX-Type APIs | APIs by category

---

qkrb_free_spnego_target_item()—Release Storage Associated with a Target Token Item

Syntax

```
#include <qkrbspnego.h>

OM_uint32 qkrb_free_spnego_target_item(
    qkrb_spnego_target_item_t **    parsed_token);
```

Service Program Name: QSYS/QKRBSPNEGO
Default Public Authority: *USE
Threadsafe: Yes

The **qkrb_free_spnego_target_item()** function releases storage associated with a qkrb_spnego_target_item_t object.

Authorities

No authorities are required.

Parameters

parsed_token (Input/Output)

The qkrb_spnego_target_item_t to be released. Upon successful completion, the *parsed_token* value is set to NULL.

Return Value

The return value is one of the following status codes:

GSS_SPNEGO_SUCCESS (0)

The routine completed successfully.

GSS_SPNEGO_UNEXPECTED_ERR (1)

The routine failed for unexpected reasons. Check the joblog for errors.

API introduced: V5R4

Top | Security APIs | UNIX-Type APIs | APIs by category

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This API descriptions publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced 36
Advanced Function Presentation
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
AS/400
BCOCA
C/400
COBOL/400
Common User Access
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
Enterprise Storage Server
eServer
FlashCopy
GDDM
i5/OS
IBM
IBM (logo)
InfoColor
Infoprint
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
Lotus
Lotus Notes
MO:DCA
MVS
Net.Data
NetServer
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
POWER5+
PowerPC
Print Services Facility
PrintManager
PROFS
RISC System/6000
RPG/400
RS/6000

SAA
SecureWay
SOM
System i
System i5
System Object Model
System/36
System/38
System/390
TotalStorage
VisualAge
WebSphere
xSeries
z/OS

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER

EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA