IBM Systems - iSeries

Network Authentication Service APIs

Version 5 Release 4

IBM Systems - iSeries

Network Authentication Service APIs

Version 5 Release 4

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 179.

Sixth Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-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

Network Authentication Service APIs

The Network Authentication Service 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. These APIs provide the means to verify the identity of a user in a network.

For more information on this topic, see Network Authentication Service.

The Network Authentication Service APIs are:

- "krb5_address_compare()—Compare Two Kerberos Addresses" on page 9 (Compare two Kerberos addresses) allows an application to compare two Kerberos addresses.
- "krb5_address_search()—Search a List of Addresses" on page 10 (Search a list of addresses) allows an application to search a list of addresses for a specific address.
- "krb5_auth_con_free()—Free an Authentication Context" on page 10 (Free an authentication context) releases an authentication context.
- "krb5_auth_con_genaddrs()—Generate Local and Remote Addresses" on page 11 (Generate local and remote addresses) generates local and remote network addresses from a socket descriptor and places them in an authentication context.
- "krb5_auth_con_getaddrs()—Get Local and Remote Addresses" on page 13 (Get local and remote addresses) retrieves the local and remote network addresses from the authentication context.
- "krb5_auth_con_getauthenticator()—Get Authenticator" on page 14 (Get authenticator) retrieves the authenticator from the authentication context.
- "krb5_auth_con_getflags()—Get Current Authentication Context Flags" on page 15 (Get current authentication context flags) retrieves the current authentication context flags.
- "krb5_auth_con_getivector()—Get Address of the Initial Vector" on page 16 (Get address of the initial vector) returns the address of the initial vector used by the specified authentication context.
- "krb5_auth_con_getkey()—Get Current Encryption Key" on page 17 (Get current encryption key) retrieves the current encryption key stored in the authentication context.
- "krb5_auth_con_getlocalseqnumber()—Get Local Message Sequence Number" on page 18 (Get local message sequence number) retrieves the local message sequence number from the authentication context.
- "krb5_auth_con_getlocalsubkey()—Get Local Subsession Key" on page 19 (Get local subsession key) retrieves the local subsession key stored in the authentication context.
- "krb5_auth_con_getports()—Get Local and Remote Network Ports" on page 19 (Get local and remote network ports) retrieves the local and remote network ports stored in the authentication context.
- "krb5_auth_con_getrcache()—Get Replay Cache Handle" on page 21 (Get replay cache handle) retrieves the replay cache for the authentication context.
- "krb5_auth_con_getremoteseqnumber()—Get Remote Message Sequence Number" on page 22 (Get remote message sequence number) retrieves the remote message sequence number from the authentication context.
- "krb5_auth_con_getremotesubkey()—Get Remote Subsession Key" on page 23 (Get remote subsession key) retrieves the remote subsession key stored in the authentication context.
- "krb5_auth_con_init()—Create and Initialize an Authentication Context" on page 23 (Create and initialize an authentication context) creates an authentication context.
- "krb5_auth_con_initvector()—Allocate and Zero the Initial Vector" on page 24 (Allocate and zero the initial vector) allocates and zeros the initial vector in the authentication context.

- "krb5_auth_con_set_req_cksumtype()—Set Checksum Type Used to Generate an Application Request Message" on page 31 (Set checksum type used to generate an application request message) sets the checksum type that will be used by the krb5_mk_req() to generate an application request message.
- "krb5_auth_con_set_safe_cksumtype()—Set Checksum Type Used to Generate a Signed Application Message" on page 33 (Set checksum type used to generate a signed application message) sets the checksum type used by the krb5_mk_safe() routine to generate a signed application message.
- "krb5_auth_con_setaddrs()—Set Local and Remote Addresses" on page 25 (Set local and remote addresses) sets the local and remote network address values in the authentication context.
- "krb5_auth_con_setflags()—Set Authentication Context Flags" on page 26 (Set authentication context flags) sets the authentication context flags.
- "krb5_auth_con_setivector()—Set Initial Vector" on page 28 (Set initial vector) sets the initial vector in the authentication context.
- "krb5_auth_con_setports()—Set Local and Remote Ports" on page 29 (Set local and remote ports) sets the local and remote network ports in the authentication context.
- "krb5_auth_con_setrcache()—Set Replay Cache Handle" on page 30 (Set replay cache handle) sets the replay cache for the authentication context.
- "krb5_auth_con_setuseruserkey()—Set User Key" on page 30 (Set user key) sets the user key in the authentication context.
- "krb5_auth_to_rep()—Convert a Kerberos Authenticator" on page 34 (Convert a Kerberos authenticator) extracts information from ticket authentication data and builds a replay cache entry.
- "krb5_build_principal()—Build a Kerberos Principal" on page 35 (Build a Kerberos principal) builds a Kerberos principal from its component strings.
- "krb5_build_principal_ext()—Build a Kerberos Principal Extended" on page 36 (Build a Kerberos principal extended) builds a Kerberos principal from its component strings.
- "krb5_build_principal_ext_va()—Build a Kerberos Principal Extended With Variable Argument List" on page 37 (Build a Kerberos principal extended with variable argument list) builds a Kerberos principal from its component strings.
- "krb5_build_principal_va()—Build a Kerberos Principal With Variable Argument List" on page 38 (Build a Kerberos principal with variable argument list) builds a Kerberos principal from its component strings.
- "krb5_cc_close()—Close a Credentials Cache" on page 40 (Close a credentials cache) closes a credentials cache.
- "krb5_cc_default()—Resolve Default Credentials Cache" on page 41 (Resolve default credentials cache) resolves the default credentials cache and returns a handle that can be used to access the cache.
- "krb5_cc_default_name()—Get Name of the Default Credentials Cache" on page 41 (Get name of the default credentials cache) returns the name of the default credentials cache for the current user.
- "krb5_cc_destroy()—Close and Delete Credentials Cache" on page 42 (Close and delete credentials cache) closes and deletes a credentials cache.
- "krb5_cc_end_seq_get()—End Sequential Reading From a Credentials Cache" on page 43 (End sequential reading from a credentials cache) unlocks the credentials cache and releases the cursor, thus ending the sequential reading of the credentials cache.
- "krb5_cc_generate_new()—Create a New Credentials Cache" on page 44 (Create a new credentials cache) creates a new credentials cache with a unique name.
- "krb5_cc_get_name()—Get Credentials Cache Name" on page 45 (Get credentials cache name) returns the name of the credentials cache.
- "krb5_cc_get_principal()—Get Principal From a Credentials Cache" on page 46 (Get principal from a credentials cache) returns the principal associated with the credentials cache.
- "krb5_cc_get_type()—Get Credentials Cache Type" on page 47 (Get credentials cache type) returns the credentials cache type.

- "krb5_cc_initialize()—Initialize Credentials Cache" on page 47 (Initialize credentials cache) initializes a credentials cache.
- "krb5_cc_next_cred()—Get Next Entry From a Credentials Cache" on page 49 (Get next entry from a credentials cache) reads the next entry from the credentials cache and returns it to the application.
- "krb5_cc_register()—Define New Credentials Cache Type" on page 49 (Define new credentials cache type) allows an application to define a new credentials cache type.
- "krb5_cc_remove_cred()—Remove Entry" on page 50 (Remove entry) removes matching entries from the credentials cache.
- "krb5_cc_resolve()—Resolve Credentials Cache Name" on page 52 (Resolve credentials cache name) resolves a credentials cache name and returns a handle that can be used to access the cache.
- "krb5_cc_retrieve_cred()—Retrieve a Set of Credentials" on page 53 (Retrieve a set of credentials) searches the credentials cache and returns an entry that matches the credentials specified.
- "krb5_cc_set_default_name()—Set Default Credentials Cache Name" on page 55 (Set Default Credentials Cache Name) sets the name of the default credentials cache for the Kerberos context.
- "krb5_cc_set_flags()—Set Credentials Cache Processing Flags" on page 55 (Set credentials cache processing flags) sets the processing flags for the credentials cache.
- "krb5_cc_start_seq_get()—Start Sequentially Retrieving Entries from a Credentials Cache" on page 57 (Start sequentially retrieving entries from a credentials cache) starts sequentially retrieving entries from the credentials cache.
- "krb5_cc_store_cred()—Store New Set of Credentials" on page 58 (Store new set of credentials) stores a new set of Kerberos credentials in the credentials cache.
- "krb5_change_password()—Change Password" on page 59 (Change Password) changes the password for the principal identified by the supplied credentials.
- "krb5_copy_address()—Copy a Kerberos Address to a New Structure" on page 60 (Copy a Kerberos address to a new structure) copies a Kerberos address to a new structure.
- "krb5_copy_addresses()—Copy an Array of Kerberos Addresses" on page 61 (Copy an array of Kerberos addresses) copies an array of Kerberos address structures.
- "krb5_copy_authdata()—Copy an Array of Authorization Data Structures" on page 62 (Copy an array of authorization data structures) copies an array of authorization data structures.
- "krb5_copy_authenticator()—Copy a Kerberos Authenticator" on page 62 (Copy a Kerberos authenticator) copies a Kerberos authenticator.
- "krb5_copy_checksum()—Copy a Kerberos Checksum" on page 63 (Copy a Kerberos checksum) copies a Kerberos checksum.
- "krb5_copy_creds()—Copy Kerberos Credentials" on page 64 (Copy Kerberos credentials) copies Kerberos credentials.
- "krb5_copy_data()—Copy a Kerberos Data Object" on page 65 (Copy a Kerberos data object) copies a Kerberos data object that is represented by a krb5_data structure.
- "krb5_copy_keyblock()—Copy a Kerberos Keyblock" on page 65 (Copy a Kerberos keyblock) copies a Kerberos keyblock.
- "krb5_copy_keyblock_contents()—Copy Contents of a Kerberos Keyblock" on page 66 (Copy contents of a Kerberos keyblock) copies the contents of a Kerberos keyblock into an existing keyblock.
- "krb5_copy_principal()—Copy a Kerberos Principal" on page 67 (Copy a Kerberos principal) copies a Kerberos principal.
- "krb5_copy_ticket()—Copy a Kerberos Ticket" on page 68 (Copy a Kerberos ticket) copies a Kerberos ticket.
- "krb5_free_address()—Free Storage Assigned to a Kerberos Address" on page 69 (Free storage assigned to a Kerberos address) releases the storage assigned to the contents of a krb5_address structure and then releases the krb5_address structure itself.

- "krb5_free_addresses()—Free Storage Assigned to Array of Kerberos Addresses" on page 69 (Free storage assigned to array of Kerberos addresses) releases the storage assigned to an array of krb5_address structures.
- "krb5_free_ap_rep_enc_part()—Free Storage Assigned to AP_REP Message Encrypted Part" on page 70 (Free storage assigned to AP_REP message encrypted part) releases the storage assigned to the decrypted portion of an AP_REP message.
- "krb5_free_authdata()—Free Storage Assigned to Array of Authentication Data" on page 71 (Free storage assigned to array of authentication data) releases the storage assigned to an array of krb5_authdata structures.
- "krb5_free_authenticator()—Free Storage Assigned to Authenticator" on page 72 (Free storage assigned to authenticator) releases the storage assigned to the contents of a krb5_authenticator structure and then releases the krb5_authenticator structure itself.
- "krb5_free_authenticator_contents()—Free Storage Assigned to Contents of Authenticator" on page 72 (Free storage assigned to contents of authenticator) releases the storage assigned to the contents of a krb5_authenticator structure.
- "krb5_free_checksum()—Free Storage Assigned to Checksum" on page 73 (Free storage assigned to checksum) releases the storage assigned to a krb5_checksum structure and then releases the krb5_checksum structure itself.
- "krb5_free_cksumtypes()—Free Checksum Types" on page 74 (Free Checksum Types) releases storage assigned to an array of checksum types.
- "krb5_free_context()—Free Kerberos Context" on page 75 (Free Kerberos context) releases a context that was created by the krb5_init_context() routine.
- "krb5_free_cred_contents()—Free Storage Assigned to Contents of a Credential" on page 76 (Free storage assigned to contents of a credential) releases the storage assigned to the contents of a krb5_creds structure.
- "krb5_free_creds()—Free Storage Assigned to a Credential" on page 75 (Free storage assigned to a credential) releases the storage assigned to the contents of a krb5_creds structure and then releases the krb5_creds structure itself.
- "krb5_free_data()—Free Storage Assigned to a Kerberos Data Object" on page 77 (Free storage assigned to a Kerberos data object) releases the storage assigned to a Kerberos data object represented by a krb5_data structure.
- "krb5_free_data_contents()—Free Storage Assigned to Contents of a Kerberos Data Object" on page 77 (Free storage assigned to contents of a Kerberos data object) releases the storage assigned to the contents of a Kerberos data object represented by a krb5_data structure.
- "krb5_free_enc_tkt_part()—Free Storage Assigned to Encrypted Ticket Part" on page 79 (Free storage assigned to encrypted ticket part) releases the storage assigned to to the krb5_enc_tkt_part structure and then releases the krb5_enc_tkt_part structure itself.
- "krb5_free_enctypes()—Free Storage Assigned to Array of Encryption Types" on page 78 (Free storage assigned to array of encryption types) releases the storage assigned to an array of encryption types.
- "krb5_free_error()—Free Storage Assigned to Kerberos Error Message" on page 80 (Free storage assigned to Kerberos error message) releases the storage assigned to the krb5_error structure and then releases the krb5_error structure itself.
- "krb5_free_host_realm()—Free Storage Assigned to Realm List" on page 80 (Free storage assigned to realm list) releases the storage assigned to a realm list.
- "krb5_free_kdc_rep()—Free Storage Assigned to KDC Reply" on page 81 (Free storage assigned to KDC reply) releases the contents of the krb5_kdc_rep structure and then releases the krb5_kdc_rep structure itself.
- "krb5_free_keyblock()—Free Storage Assigned to a Keyblock" on page 82 (Free storage assigned to a keyblock) releases the contents of the krb5_keyblock structure and then releases the krb5_keyblock structure itself.

- "krb5_free_keyblock_contents()—Free Storage Assigned to Contents of a Keyblock" on page 83 (Free storage assigned to contents of a keyblock) releases the contents of the krb5_keyblock structure.
- "krb5_free_krbhst()—Free Storage Assigned to Host List" on page 83 (Free storage assigned to host list) releases the storage assigned to a host list.
- "krb5_free_principal()—Free Storage Assigned to Principal" on page 84 (Free storage assigned to principal) releases the storage assigned to a krb_5 principal.
- "krb5_free_string()—Free Storage Assigned to Character String" on page 85 (Free storage assigned to character string) releases the storage assigned to a character string.
- "krb5_free_tgt_creds()—Free Storage Assigned to Array of Credentials" on page 85 (Free storage assigned to array of credentials) releases the storage assigned to an array of krb5_creds structures.
- "krb5_free_ticket()—Free Storage Assigned to a Ticket" on page 86 (Free storage assigned to a ticket) releases the storage assigned to a krb5_ticket structure and then releases the krb5_ticket structure itself.
- "krb5_free_tickets()—Free Storage Assigned to Array of Tickets" on page 87 (Free storage assigned to array of tickets) releases the storage assigned to an array of krb5_ticket structures.
- "krb5_gen_replay_name()—Generate Replay Cache Name" on page 89 (Generate replay cache name) generates a unique replay cache name based on the Kerberos address supplied by the caller.
- "krb5_generate_seq_number()—Generate Random Sequence Number" on page 88 (Generate random sequence number) generates a random sequence number based on the supplied key.
- "krb5_generate_subkey()—Generate Subsession Key" on page 88 (Generate subsession key) generates a random subsession key that is based on the supplied session key.
- "krb5_get_cred_from_kdc()—Get Service Ticket from Kerberos KDC Server" on page 94 (Get service ticket from Kerberos KDC server) obtains a service ticket from the Kerberos Key Distribution Center (KDC) server.
- "krb5_get_cred_from_kdc_renew()—Renew Service Ticket Obtained from Kerberos KDC Server" on page 95 (Renew service ticket obtained from Kerberos KDC server) renews a service ticket obtained from the Kerberos Key Distribution Center (KDC) server.
- "krb5_get_cred_from_kdc_validate()—Validate Service Ticket Obtained from Kerberos KDC Server" on page 97 (Validate service ticket obtained from Kerberos KDC server) validates a service ticket obtained from the Kerberos Key Distribution Center (KDC) server.
- "krb5_get_cred_via_tkt()—Get Service Ticket from Kerberos KDC Server Using Supplied Ticket-granting Ticket" on page 98 (Get service ticket from Kerberos KDC server using supplied ticket-granting ticket) obtains a service ticket from the Kerberos Key Distribution Center (KDC) server.
- "krb5_get_credentials()—Get Service Ticket" on page 90 (Get service ticket) obtains a service ticket for the requested server.
- "krb5_get_credentials_renew()—Renew Service Ticket" on page 92 (Renew service ticket) renews a service ticket for the rquested service.
- "krb5_get_credentials_validate()—Validate Service Ticket" on page 93 (Validate service ticket) validates a service ticket for the requested service.
- "krb5_get_default_in_tkt_ktypes()—Get Default Encryption Types to be Used for Initial Ticket" on page 100 (Get default encryption types to be used for initial ticket) returns the default encryption types that are used when requesting an initial ticket from the Kerberos server.
- "krb5_get_default_realm()—Get Default Realm" on page 100 (Get default realm) returns the default realm for the local system.
- "krb5_get_default_tgs_ktypes()—Get Default Encryption Types to be Used for Service Ticket" on page 101 (Get default encryption types to be used for service ticket) returns the default encryption types that are used when requesting a service ticket from the Kerberos server.
- "krb5_get_host_realm()—Get Kerberos Realm Name for Host Name" on page 102 (Get Kerberos realm name for host name) returns a list of Kerberos realm names for the specified host name.
- "krb5_get_in_tkt_with_keytab()—Get Initial Ticket Using Key Table" on page 103 (Get initial ticket using key table) obtains an initial ticket-granting ticket from the Kerberos Key Distribution Center (KDC) server using a key table.

- "krb5_get_in_tkt_with_password()—Get Initial Ticket Using Text Password" on page 105 (Get initial ticket using text password) obtains an initial ticket-granting ticket from the Kerberos Key Distribution Center (KDC) server using a text password.
- "krb5_get_in_tkt_with_skey()—Get Initial Ticket Using Session Key" on page 107 (Get initial ticket using session key) obtains an initial ticket-granting ticket from the Kerberos Key Distribution Center (KDC) server using a session key.
- "krb5_get_krbhst()—Get List of KDC Hosts" on page 109 (Get list of KDC hosts) returns a list of Kerberos Key Distribution Center (KDC) server hosts for a Kerberos realm.
- "krb5_get_server_rcache()—Generate Replay Cache for Server Use" on page 110 (Generate replay cache for server use) generates a unique replay cache name and then opens the replay cache.
- "krb5_init_context()—Create and Initialize a Kerberos Context" on page 111 (Create and initialize a Kerberos context) creates a new Kerberos context and initializes it with default values obtained from the Kerberos configuration file.
- "krb5_kt_add_entry()—Add New Entry to Key Table" on page 112 (Add new entry to key table) adds a new entry to a key table.
- "krb5_kt_close()—Close Key Table" on page 113 (Close key table) closes a key table.
- "krb5_kt_default()—Resolve Default Key Table" on page 114 (Resolve default key table) resolves the default key table and returns a handle that can be used to access the table.
- "krb5_kt_default_name()—Get Default Key Table Name" on page 115 (Get default key table name) returns the name of the default key table for the current user.
- "krb5_kt_end_seq_get()—End Sequential Reading of Key Table" on page 116 (End sequential reading of key table) ends the sequential reading of the key table and releases the cursor.
- "krb5_kt_free_entry()—Free Storage Assigned to Key Table Entry" on page 117 (Free storage assigned to key table entry) releases the storage assigned to a key table entry.
- "krb5_kt_get_entry()—Get Entry from Key Table" on page 117 (Get entry from key table) returns an entry from the key table.
- "krb5_kt_get_name()—Get Key Table Name" on page 118 (Get key table name) returns the name of the key table in the application-provided buffer supplied in the name parameter.
- "krb5_kt_get_type()—Get Key Table Type" on page 119 (Get key table type) returns the key table type.
- "krb5_kt_next_entry()—Get Next Entry from Key Table" on page 120 (Get next entry from key table) reads the next entry from the key table and returns it to the application.
- "krb5_kt_read_service_key()—Get Service Key from Key Table" on page 121 (Get service key from key table) returns the service key from the key table.
- "krb5_kt_register()—Register New Key Table Type" on page 122 (Register new key table type) registers a new key table type.
- "krb5_kt_remove_entry()—Remove Entry from Key Table" on page 123 (Remove entry from key table) removes an entry from a key table.
- "krb5_kt_resolve()—Resolve Key Table Name" on page 124 (Resolve key table name) resolves a key table name and returns a handle that can be used to access the table.
- "krb5_kt_start_seq_get()—Start Sequentially Retrieving Entries from Key Table" on page 125 (Start sequentially retrieving entries from key table) starts sequentially retrieving entries from the key table.
- "krb5_md5_crypto_compat_ctl()—Set Compatibility Mode for MD5 Checksum Generation" on page 126 (Set compatibility mode for MD5 checksum generation) sets the compatibility mode for the MD5 DES checksum generation.
- "krb5_mk_error()—Create Kerberos KRB_ERROR Message" on page 127 (Create Kerberos KRB_ERROR message) creates a Kerberos KRB_ERROR message.
- "krb5_mk_priv()—Create Kerberos KRB_PRIV Message" on page 128 (Create Kerberos KRB_PRIV message) creates a Kerberos KRB_PRIV message using data supplied by the application.
- "krb5_mk_rep()—Create Kerberos AP_REP Message" on page 129 (Create Kerberos AP_REP message) creates a Kerberos AP_REP message using information in the authentication context.

- "krb5_mk_req()—Create Kerberos AP_REQ Message" on page 130 (Create Kerberos AP_REQ message) creates a Kerberos AP_REQ message.
- "krb5_mk_req_extended()—Create Kerberos AP_REQ Message Using Supplied Credentials" on page 132 (Create Kerberos AP_REQ message using supplied credentials) creates a Kerberos AP_REQ message using supplied credentials.
- "krb5_mk_safe()—Create Kerberos KRB_SAFE Message" on page 133 (Create Kerberos KRB_SAFE message) creates a Kerberos KRB_SAFE message using data supplied by the application.
- "krb5_os_hostaddr()—Get Network Addresses Used by Specific Host System" on page 135 (Get network addresses used by specific host system) returns the network addresses used by a specific host system.
- "krb5_os_localaddr()—Return Network Addresses Used by Local System" on page 136 (Return network addresses used by local system) returns the network addresses used by the local system.
- "krb5_parse_name()—Create Kerberos Principal from Text String" on page 137 (Create Kerberos principal from text string) converts a text string into a Kerberos principal.
- "krb5_principal_compare()—Compare Two Kerberos Principals" on page 138 (Compare two Kerberos principals) allows an application to compare two Kerberos principals.
- "krb5_random_confounder()—Create Random Confounder" on page 138 (Create random confounder) creates a random value that can be used as a confounder when encrypting data.
- "krb5_rc_close()—Close Replay Cache" on page 139 (Close replay cache) closes a replay cache.
- "krb5_rc_default()—Resolve Default Replay Cache" on page 140 (Resolve default replay cache) resolves the default replay cache and returns a handle that can be used to access the table.
- "krb5_rc_default_name()—Get Default Replay Cache Name" on page 141 (Get default replay cache name) returns the name of the default replay cache for the current user.
- "krb5_rc_destroy()—Delete Replay Cache" on page 141 (Delete replay cache) closes and deletes a replay cache.
- "krb5_rc_expunge()—Delete Expired Entries from Replay Cache" on page 143 (Delete expired entries from replay cache) deletes expired entries from the replay cache.
- "krb5_rc_free_entry_contents()—Free Storage Associated with Replay Cache Entry" on page 143 (Free storage associated with replay cache entry) releases the storage associated with a replay cache entry.
- "krb5_rc_get_lifespan()—Get Authenticator Lifespan for Entries in Replay Cache" on page 144 (Get authenticator lifespan for entries in replay cache) returns the authenticator lifespan for entries in the replay cache.
- "krb5_rc_get_name()—Get Replay Cache Name" on page 145 (Get replay cache name) returns the replay cache name.
- "krb5_rc_get_type()—Get Replay Cache Type" on page 146 (Get replay cache type) returns the replay cache type.
- "krb5_rc_initialize()—Initialize Replay Cache" on page 146 (Initialize replay cache) initializes a replay cache.
- "krb5_rc_recover()—Recover Replay Cache" on page 147 (Recover replay cache) recovers a replay cache after the application has been restarted.
- "krb5_rc_register_type()—Define New Replay Cache Type" on page 148 (Define new replay cache type) allows an application to define a new replay cache type.
- "krb5_rc_resolve()—Resolve Replay Cache Name" on page 149 (Resolve replay cache name) resolves a replay cache name and returns a handle that can be used to access the cache.
- "krb5_rc_store()—Store New Entry in Replay Cache" on page 150 (Store new entry in replay cache) stores a new entry in the replay cache after verifying that the entry is not already in the cache.
- "krb5_rd_error()—Process Kerberos KRB_ERROR Message" on page 151 (Process Kerberos KRB_ERROR message) processes a Kerberos KRB_ERROR message created by the krb5_mk_error() routine and returns a krb5_error structure.

- "krb5_rd_priv()—Process Kerberos KRB_PRIV Message" on page 152 (Process Kerberos KRB_PRIV message) processes a Kerberos KRB_PRIV message and extracts the application data after verifying its integrity.
- "krb5_rd_rep()—Process Kerberos AP_REP Message" on page 154 (Process Kerberos AP_REP message) processes a Kerberos AP_REP message created by the krb5_mk_rep() routine.
- "krb5_rd_req()—Process Kerberos AP_REQ Message" on page 155 (Process Kerberos AP_REQ message) processes a Kerberos AP_REQ message generated by the partner application.
- "krb5_rd_req_verify()—Process and Verify Kerberos AP_REQ Message" on page 156 (Process and Verify Kerberos AP_REQ Message) processes an AP_REQ message generated by the partner application and verifies the application data checksum contained in the authenticator.
- "krb5_rd_safe()—Process Kerberos KRB_SAFE Message" on page 158 (Process Kerberos KRB_SAFE message) processes a Kerberos KRB_SAFE message and extracts the application data after verifying its integrity.
- "krb5_realm_compare()—Compare Realm Names of Two Principals" on page 159 (Compare realm names of two principals) compares the realm names of two principals.
- "krb5_recvauth()—Process an Authentication Message Stream" on page 160 (Process an Authentication Message Stream) processes an authentication message stream generated by the**krb5_sendauth()**routine.
- "krb5_sendauth()—Send an Authentication Message Stream" on page 162 (Send an Authentication Message Stream) generates an authentication message stream for processing by the**krb5_recvauth()**routine.
- "krb5_set_config_files()—Set Files to be Processed for Kerberos Configuration Requests" on page 164 (Set files to be processed for Kerberos configuration requests) specifies the names of the files to be processed to obtain the Kerberos configuration.
- "krb5_set_default_in_tkt_ktypes()—Set Default Encryption Types to Request Initial Ticket" on page 165 (Set default encryption types to request initial ticket) sets the default encryption types to be used when requesting an initial ticket from the Kerberos server.
- "krb5_set_default_realm()—Set Default Realm for Local System" on page 166 (Set default realm for local system) sets the default realm for the specified Kerberos context.
- "krb5_set_default_tgs_ktypes()—Set Default Encryption Types to Request Service Ticket" on page 167 (Set default encryption types to request service ticket) sets the default encryption types to be used when requesting a service ticket from the Kerberos server.
- "krb5_sname_to_principal()—Convert Service Name to a Kerberos Principal" on page 168 (Convert service name to a Kerberos principal) converts a service name and a host name to a Kerberos principal.
- "krb5_svc_get_msg()—Get Printable Text Message Corresponding to Kerberos Error Code" on page 169 (Get printable text message corresponding to Kerberos error code) returns a printable text message corresponding to a Kerberos error code.
- "krb5_timeofday()—Get Current Time of Day in Seconds since the Epoch" on page 170 (Get current time of day in seconds since the epoch) returns the current time of day in seconds since the epoch (January 1, 1970).
- "krb5_unparse_name()—Convert a Kerberos Principal to Text String" on page 171 (Convert a Kerberos principal to text string) creates a text string from a Kerberos principal.
- "krb5_unparse_name_ext()—Convert a Kerberos Principal Extended to Text String" on page 172 (Convert a Kerberos principal extended to text string) creates a text string from a Kerberos principal.
- "krb5_us_timeofday()—Get Current Time of Day in Seconds and Microseconds since the Epoch" on page 173 (Get current time of day in seconds and microseconds since the epoch) returns the current time of day in seconds and microseconds since the epoch (January 1, 1970).
- "qkrb_add_kt_entry()—Add Keytab Entry" on page 174 (Add Keytab Entry) allows you to add a keytab entry to a keytab file for a specified principal name.
- "qkrb_count_kt_entries()—Count Keytab Entries" on page 175 (Count Keytab Entries) allows you to obtain the total count of entries in a keytab file or count the number of keytab entries there are for a particular principal.

- "qkrb_remove_kt_entry()—Remove Keytab Entry" on page 177 (Remove Keytab Entry) allows you to remove keytab entries from a keytab file for a specified principal

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

APIs

These are the APIs for this category.

krb5_address_compare()—Compare Two Kerberos Addresses

Syntax

```
#include <krb5.h>

krb5_boolean krb5_address_compare(
    krb5_context  context,
    krb5_const krb5_address * addr1,
    krb5_const krb5_address * addr2);
```

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

The **krb5_address_compare()** function allows an application to compare two Kerberos addresses.

Parameters

context (Input)
The Kerberos context.

addr1 (Input)
The first address.

addr2 (Input)
The second address.

Return Value

TRUE	The addresses are the same.
FALSE	The addresses are different.

Authorities

No authorities are required.

Error Messages

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

API introduced: V5R1

Top | Security APIs | APIs by category

krb5_address_search()—Search a List of Addresses

Syntax

```
#include <krb5.h>

krb5_boolean krb5_address_search(
    krb5_context     context,
    krb5_const krb5_address *    addr,
    krb5_address * krb5_const *  addrlist);
```

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

The **krb5_address_search()** function allows an application to search a list of addresses for a specific address.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

addr (Input)
The search address.

addrlist (Input)
The address list as an array of addresses. The last entry in the array must be a **NULL** pointer. Specify **NULL** for this parameter if no address list is present.

Return Value

TRUE	The search address was found in the address list, or the address list was not provided.
FALSE	The search address was not found in the address list.

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

krb5_auth_con_free()—Free an Authentication Context

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_free(
    krb5_context     context,
    krb5_auth_context      auth_context);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_free()** function releases an authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

---

krb5_auth_con_genaddrs()—Generate Local and Remote Addresses

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_genaddrs(
    krb5_context     context,
    krb5_auth_context      auth_context,
    int        fd,
    int        flags);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_genaddrs()** function generates local and remote network addresses from a socket descriptor and places them in an authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

fd (Input)
The socket descriptor to be used.

flags (Input)
The address generation flags as follows:

KRB5_AUTH_CONTEXT_GENERATE_LOCAL_ADDR (x'00000001')	Generate the local network address.
KRB5_AUTH_CONTEXT_GENERATE_LOCAL_FULL_ADDR ('x00000004')	Generate the local network address and the local port.
KRB5_AUTH_CONTEXT_GENERATE_REMOTE_ADDR (x'00000002')	Generate the remote network address.
KRB5_AUTH_CONTEXT_GENERATE_REMOTE_FULL_ADDR (x'00000008')	Generate the remote network address and the remote port.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The addresses generated by this routine can be retrieved by the application by calling **krb5_auth_con_getaddrs()** and **krb5_auth_con_getports()**.
2. The socket must have been created using the **AF_INET** address family. The socket must be in the connected state if the remote network address is to be generated.
3. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_getaddrs()—Get Local and Remote Addresses

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getaddrs(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_address **      local_addr,
    krb5_address **      remote_addr);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getaddrs()** function retrieves the local and remote network addresses from the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

local_addr (Output)
The local network address. Specify **NULL** for this parameter if the local network address is not required. The returned value is **NULL** if the local network address has not been set. The **krb5_free_address()** routine should be called to release the address when it is no longer needed.

remote_addr (Output)
The remote network address. Specify **NULL** for this parameter if the remote network address is not required. The return value is **NULL** if the remote network address has not been set. The **krb5_free_address()** routine should be called to release the address when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_auth_con_getauthenticator()—Get Authenticator

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getauthenticator(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_authenticator **    authent);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getauthenticator()** function retrieves the authenticator from the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

authent (Output)
The authenticator. The **krb5_free_authenticator()** routine should be called to release the authenticator when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_auth_con_getflags()—Get Current Authentication Context Flags

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getflags(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_int32 *     flags);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getflags()** function retrieves the current authentication context flags.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

flags (Output)
The current flags. The following symbolic definitions are provided for the flag bits:

KRB5_AUTH_CONTEXT_DO_TIME (x'00000001')	Use timestamps in messages.
KRB5_AUTH_CONTEXT_RET_TIME (x'00000002')	Return timestamps to the application.
KRB5_AUTH_CONTEXT_DO_SEQUENCE (x'00000004')	Use sequence numbers in messages.
KRB5_AUTH_CONTEXT_RET_SEQUENCE (x'00000008')	Return sequence numbers to the application.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_auth_con_getivector()—Get Address of the Initial Vector

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getivector(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_pointer *     ivec);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getivector()** routine returns the address of the initial vector used by the specified authentication context. The application can then use this address to change the contents of the initial vector. The application, however, must not free the storage represented by the initial vector.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

ivec (Output)
The address of the initial vector. The authentication context still points to this vector, so any changes made to the vector may affect future data encryption operations performed using the authentication context.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_auth_con_getkey()—Get Current Encryption Key

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getkey(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_keyblock **     keyblock);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getkey()** routine retrieves the current encryption key stored in the authentication context. Normally, this is the session key that was obtained from an application request message.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

keyblock (Output)
A keyblock containing the encryption key. The **krb5_free_keyblock()** routine should be called to release the keyblock when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

CPE3418 E Possible APAR condition or hardware failure.

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_auth_con_getlocalseqnumber()—Get Local Message Sequence Number

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getlocalseqnumber(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_int32 *     seqnum);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getlocalseqnumber()** function retrieves the local message sequence number from the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

seqnum (Output)
The message sequence number.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. Sequence numbers are used when generating messages if the **KRB5_AUTH_CONTEXT_DO_SEQUENCE** (x'00000004') flag has been set in the authentication context.
2. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_getlocalsubkey()—Get Local Subsession Key

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getlocalsubkey(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_keyblock **     keyblock);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getlocalsubkey()** function retrieves the local subsession key stored in the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

keyblock (Output)
A keyblock containing the subsession key. The **krb5_free_keyblock()** routine should be called to release the keyblock when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_getports()—Get Local and Remote Network Ports

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getports(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_address **      local_port,
    krb5_address **      remote_port);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getports()** function retrieves the local and remote network ports stored in the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

local_port (Output)
The local network port. Specify **NULL** for this parameter if the local network port is not required. The return value is **NULL** if the local network port has not been set. The **krb5_free_address()** routine should be called to release the address when it is no longer needed.

remote_port (Output)
The remote network port. Specify **NULL** for this parameter if the remote network port is not required. The return value is **NULL** if the remote network port has not been set. The **krb5_free_address()** routine should be called to release the address when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_getrcache()—Get Replay Cache Handle

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getrcache(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_rcache *      rcache);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getrcache()** function retrieves the replay cache for the authentication context. A replay cache is used to detect message replay when processing a message. A replay cache must be set in the authentication context if message timestamps are being used.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

rcache (Output)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_getremoteseqnumber()—Get Remote Message Sequence Number

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getremoteseqnumber(
    krb5_context   context,
    krb5_auth_context   auth_context,
    krb5_int32 *   seqnum);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_getremoteseqnumber()** function retrieves the remote message sequence number from the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
Tthe authentication context.

seqnum (Output)
The message sequence number.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. Sequence numbers are used when generating messages if the **KRB5_AUTH_CONTEXT_DO_SEQUENCE** (x'00000004') flag has been set in the authentication context.
2. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_getremotesubkey()—Get Remote Subsession Key

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_getremotesubkey(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_keyblock **     keyblock);
```

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

The **krb5_auth_con_getremotesubkey()** function retrieves the remote subsession key stored in the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

keyblock (Output)
A keyblock containing the subsession key. The **krb5_free_keyblock()** routine should be called to release the keyblock when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_init()—Create and Initialize an Authentication Context

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_init(
    krb5_context     context,
    krb5_auth_context *    auth_context);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_init()** function creates an authentication context. An authentication context contains information relating to a single connection between two applications. The context is initialized to enable the use of the replay cache (**KRB5_AUTH_CONTEXT_DO_TIME** (x'00000001')), but to disable the use of message sequence numbers. The **krb5_auth_con_setflags()** routine can be used to change these defaults.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Output)
The authentication context created by this call. The **krb5_auth_con_free()** routine should be called to release the authentication context when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_auth_con_initvector()—Allocate and Zero the Initial Vector

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_initvector(
    krb5_context     context,
    krb5_auth_context      auth_context);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_initivector()** function allocates and zeros the initial vector in the authentication context. The authentication context must already contain an encryption key defining the type of encryption that will be used. The initial vector is used to initialize the encryption sequence each time a message is encrypted. This serves to generate different encrypted results for the same message contents and encryption key.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The application should not use both **krb5_auth_con_initivector()** and **krb5_auth_con_setivector()** for the same authentication context.
2. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

---

krb5_auth_con_setaddrs()—Set Local and Remote Addresses

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_setaddrs(
```

```
krb5_context     context,
krb5_auth_context      auth_context,
krb5_address *     local_addr,
krb5_address *     remote_addr);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_setaddrs()** function sets the local and remote network address values in the authentication context. These values are used when obtaining tickets and constructing authenticators.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

local_addr (Input)
The local network address. Specify **NULL** for this parameter if the local network address is not to be changed.

remote_addr (Input)
The remote network address. Specify **NULL** for this parameter if the remote network address is not to be changed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

---

krb5_auth_con_setflags()—Set Authentication Context Flags

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_setflags(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_int32       flags);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_setflags()** function sets the authentication context flags.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

flags (Input)
The current flags. The following symbolic definitions are provided for the flag bits:

KRB5_AUTH_CONTEXT_DO_TIME (x'00000001')	Use timestamps in messages.
KRB5_AUTH_CONTEXT_RET_TIME (x'00000002')	Return timestamps to the application.
KRB5_AUTH_CONTEXT_DO_SEQUENCE (x'00000004')	Use sequence numbers in messages.
KRB5_AUTH_CONTEXT_RET_SEQUENCE (x'00000008')	Return sequence numbers to the application.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_setivector()—Set Initial Vector

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_setivector(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_pointer       ivec);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_setivector()** function sets the initial vector in the authentication context. A copy is not made of the initial vector, so the application must not change or free the buffer specified by the *ivec* parameter until either a new initial vector is set or the authentication context is released. The initial vector is used to initialize the encryption sequence each time a message is encrypted. This generates different encrypted results for the same message contents and encryption key.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

ivec (Input)
The initial vector.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The application should not use both **krb5_auth_con_initivector()** and **krb5_auth_con_setivector()** for the same authentication context.
2. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_setports()—Set Local and Remote Ports

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_setports(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_address *     local_port,
    krb5_address *     remote_port);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_setports()** function sets the local and remote network ports in the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

local_port (Input)
The local network port. Specify **NULL** for this parameter if the local network port is not to be changed.

remote_port (Input)
The remote network port. Specify **NULL** for this parameter if the remote network port is not to be changed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_setrcache()—Set Replay Cache Handle

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_setrcache(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_rcache      rcache);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_setrcache()** function sets the replay cache for the authentication context. A replay cache is used to detect message replay when processing a message. A replay cache must be set in the authentication context if message timestamps are being used. The **krb5_rc_default()** and **krb5_rc_resolve()** routines can be used to obtain a replay cache handle.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

rcache (Input)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_setuseruserkey()—Set User Key

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_setuseruserkey(
    krb5_context   context,
    krb5_auth_context   auth_context,
    krb5_keyblock *    keyblock);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_setuseruserkey()** function sets the user key in the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

keyblock (Input)
The user key.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The **krb5_auth_con_setuseruserkey()** routine is only useful prior to calling the **krb5_rd_req()** routine for user-to-user authentication where the server has the key and needs to use it to decrypt the incoming request. Once the request has been decrypted, this key is no longer necessary and is replaced in the authentication context with the session key obtained from the decoded request.
2. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

---

krb5_auth_con_set_req_cksumtype()—Set Checksum Type Used to Generate an Application Request Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_set_req_cksumtype(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_cksumtype     cksumtype);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_set_req_cksumtype()** function sets the checksum type that will be used by the **krb5_mk_req()** to generate an application request message. This overrides the default value set by the *ap_req_checksum_type* entry in the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

cksumtype (Input)
The checksum type as follows:

CKSUMTYPE_CRC32 (x'0001')	DES CRC checksum
CKSUMTYPE_DESCBC (x'0004')	DES CBC checksum
CKSUMTYPE_RSA_MD5 (x'0007')	MD5 checksum
CKSUMTYPE_RSA_MD5_DES (x'0008')	DES MD5 checksum

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

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

krb5_auth_con_set_safe_cksumtype()—Set Checksum Type Used to Generate a Signed Application Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_con_set_safe_cksumtype(
    krb5_context     context,
    krb5_auth_context      auth_context,
    krb5_cksumtype     cksumtype);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_con_set_safe_cksumtype()** function sets the checksum type used by the **krb5_mk_safe()** routine to generate a signed application message. This overrides the default value set by the *safe_checksum_type* entry in the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

cksumtype (Input)
The checksum type as follows:

CKSUMTYPE_CRC32 (x'0001')	DES CRC checksum
CKSUMTYPE_DESCBC (x'0004')	DES CBC checksum
CKSUMTYPE_RSA_MD5 (x'0007')	MD5 checksum
CKSUMTYPE_RSA_MD5_DES (x'0008')	DES MD5 checksum

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_auth_to_rep()—Convert a Kerberos Authenticator

Syntax

```
#include <krb5.h>

krb5_error_code krb5_auth_to_rep(
    krb5_context     context,
    krb5_tkt_authent *     authent,
    krb5_donot_replay *    replay);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_auth_to_rep()** function extracts information from ticket authentication data and builds a replay cache entry. This entry can then be used to check for ticket replay by calling the **krb5_rc_store()** routine to save the entry in the replay cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

authent (Input)
The Kerberos authenticator.

replay (Output)
The replay entry data. The **krb5_rc_free_entry_contents()** routine should be called to release the entry data when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time.

API introduced: V5R1

krb5_build_principal()—Build a Kerberos Principal

Syntax

```
#include <krb5.h>

krb5_error_code krb5_build_principal(
    krb5_context     context,
    krb5_principal *     ret_principal,
    int        realm_length,
    krb5_const char *      realm,
    char *       name1, name2, ...);
```

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

The **krb5_build_principal()** function builds a Kerberos principal from its component strings.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ret_principal (Output)
The Kerberos principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

realm_length (Input)
The length of the realm name.

realm (Input)
The realm name.

name1, name2, ... (Input)
One or more name components. The end of the components is indicated by specifying **NULL** for the parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates the principal bambi/admin@forest:

```
#include <krb5.h>

retval = krb5_build_principal(context, &princ, 6, "forest", "bambi", "admin", NULL);
```

API introduced: V5R1

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

---

krb5_build_principal_ext()—Build a Kerberos Principal Extended

Syntax

```
#include <krb5.h>

krb5_error_code krb5_build_principal_ext(
    krb5_context     context,
    krb5_principal *     ret_principal,
    int        realm_length,
    krb5_const char *      realm,
    int        name1_len,
    char *                              name1,
    int        name2_len,,
    char *       name2, ...);
```

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

The **krb5_build_principal_ext()** function builds a Kerberos principal from its component strings. It is similar to the **krb5_build_principal()** routine, except the name component lengths are explicitly specified on the function call.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ret_principal (Output)
The Kerberos principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

realm_length (Input)
The length of the realm name.

realm (Input)
The realm name.

name1_len, name1, name2_length, name2, ... (Input)
One or more name components. Each component consists of its length followed by its value. The end of the components is indicated by specifying a length of zero.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates the principal bambi/admin@forest:

```
#include <krb5.h>

retval = krb5_build_principal_ext(context, &princ, 6, "forest", 5, "bambi", 5, "admin", 0);
```

API introduced: V5R1

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

---

krb5_build_principal_ext_va()—Build a Kerberos Principal Extended With Variable Argument List

Syntax

```
#include <stdarg.h>
#include <krb5.h>

krb5_error_code krb5_build_principal_ext_va(
    krb5_context     context,
    krb5_principal *     ret_principal,
    int        realm_length,
    krb5_const char *      realm,
    va_list        ap);
```

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

The **krb5_build_principal_ext_va()** function builds a Kerberos principal from its component strings. It is similar to the **krb5_build_principal_ext()** routine, except the name components are specified as a variable argument list instead of as discrete parameters on the function call.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ret_principal (Output)
The Kerberos principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

realm_length (Input)
The length of the realm name.

realm (Input)
The realm name.

ap (Input)
A variable argument list consisting of name lengths and character pointers that specify one or more name components. The end of the components is indicated by specifying a name length of zero.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Example

See Code disclaimer information for information pertaining to code examples.

Assume we have a function my_func that is called with a list of names. It could generate a Kerberos principal from these names as follows:

```
#include <stdarg.h>
#include <krb5.h>

krb5_error_code my_func(int realm_len, char *realm, ...) {
    va_list ap;
    krb5_error_code retval;

    va_start(ap, realm);

    retval = krb5_build_principal_ext_va(context, &princ,realm_len, realm, ap);

    va_end(ap);

    return retval;
}

int main(int argc, char *argv[]) {

    my_func(6, "forest", 5, "bambi", 5, "admin", 0);

    return 0;
}
```

API introduced: V5R1

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

---

krb5_build_principal_va()—Build a Kerberos Principal With Variable Argument List

Syntax

```
#include <stdarg.h>
#include <krb5.h>

krb5_error_code krb5_build_principal_va(
    krb5_context   context,
    krb5_principal *   ret_principal,
    int    realm_length,
    krb5_const char *   realm,
    va_list    ap);
```

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

The **krb5_build_principal_va()** function builds a Kerberos principal from its component strings. It is similar to the **krb5_build_principal()** routine, except the name components are specified as a variable argument list instead of as discrete parameters on the function call.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ret_principal (Output)
Tthe Kerberos principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

realm_length (Input)
The length of the realm name.

realm (Input)
The realm name.

ap (Input)
A variable argument list consisting of character pointers that specify one or more name components. The end of the components is indicated by specifying **NULL** for the parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

CPE3418 E Possible APAR condition or hardware failure.

Example

See Code disclaimer information for information pertaining to code examples.

Assume we have a function my_func that is called with a list of names. It could generate a Kerberos principal from these names as follows:

```
#include <stdarg.h>
#include <krb5.h>

krb5_error_code my_func(char *realm, ...) {
    va_list ap;
```

```
    krb5_error_code retval;

    va_start(ap, realm);

    retval = krb5_build_principal_va(context, &princ,strlen(realm), realm, ap);

    va_end(ap);

    return retval;
}

int main(int argc, char *argv[]) {

    my_func(6, "forest", "bambi", "admin", NULL);

    return 0;
}
```

API introduced: V5R1

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

---

krb5_cc_close()—Close a Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_close(
    krb5_context     context,
    krb5_ccache      ccache);
```

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

The **krb5_cc_close()** function closes a credentials cache. Once this function is completed, the cache handle may not be used.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_cc_default()—Resolve Default Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_default(
    krb5_context     context,
    krb5_ccache *      ccache);
```

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

The **krb5_cc_default()** function resolves the default credentials cache and returns a handle that can be used to access the cache. This is equivalent to calling the **krb5_cc_resolve()** routine with the name returned by the **krb5_cc_default_name()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Output)
The credentials cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_cc_default_name()—Get Name of the Default Credentials Cache

Syntax

```
#include <krb5.h>

char * krb5_cc_default_name(
    krb5_context     context);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See “Usage Notes” on page 42.

The **krb5_cc_default_name()** function returns the name of the default credentials cache for the current user. If the **KRB5CCNAME** environment variable is set, this is the name of the default cache. Otherwise, the name is obtained from the file specified by the **_EUV_SEC_KRB5CCNAME_FILE** environment variable. If this environment variable is not set, the name is obtained from the **krb5ccname** in the HOME directory. If this file does not exist or if there is no default credentials cache name set in the file, a new credentials cache file is created.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

Return Value

The name of the default credentials cache for the current user. This is a pointer to read-only storage and must not be freed by the application. If an error occurs, the function return value is **NULL**.

Error Messages

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

Usage Notes

1. The **krb5_cc_default_name()** routine uses static storage within the Kerberos context to hold the default name; therefore, this routine is not threadsafe unless a separate context is used for each thread.

API introduced: V5R1

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

---

krb5_cc_destroy()—Close and Delete Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_destroy(
    krb5_context     context,
    krb5_ccache      ccache);
```

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

The **krb5_cc_destroy()** function closes and deletes a credentials cache. Once this function is completed, the cache handle may not be used.

Authorities

When the credentials cache is of type "FILE" (see **krb5_cc_resolve()** for more information on cache types), the default behavior is that the credentials cache file is created in the

/QIBM/UserData/OS400/NetworkAuthentication/creds directory. The placement of the credentials cache file can be changed by setting the KRB5CCNAME environment variable.

If the credentials cache file does not reside in the default directory, the following authorities are required:

Object Referred to	Data Authority Required	Object Authority Required
Each directory in the path name preceding the credentials cache file	*X	None
Parent directory of the credentials cache file	*WX	None
Credentials cache file	*RW	*OBJEXIST

If the credentials cache file resides in the default directory, the following authorities are required:

Object Referred to	Data Authority Required	Object Authority Required
All directories in the path name	*X	None
Credentials cache file	*RW	None

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_cc_end_seq_get()—End Sequential Reading From a Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_end_seq_get(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_cc_cursor *     cursor);
```

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

The **krb5_cc_end_seq_get()** routine unlocks the credentials cache and releases the cursor, thus ending the sequential reading of the credentials cache. The cursor may not be used once **krb5_cc_end_seq_get()** has completed.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

cursor (Input/Output)
The cursor created by the **krb5_cc_start_seq_get()** routine.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_cc_generate_new()—Create a New Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_generate_new(
    krb5_context     context,
    krb5_const char *      type,
    krb5_ccache *      ccache);
```

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

The **krb5_cc_generate_new()** function creates a new credentials cache with a unique name. The **krb5_cc_initialize()** function must be called to set the cache principal before storing any credentials in the cache.

Authorities

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

Parameters

context (Input)
The Kerberos context.

type (Input)
The credentials cache type (for example, FILE).

ccache (Output)
The credentials cache handle. The **krb5_cc_close()** or **krb5_cc_destroy()** routine should be called to release the handle when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_cc_get_name()—Get Credentials Cache Name

Syntax

```
#include <krb5.h>

char * krb5_cc_get_name(
    krb5_context     context,
    krb5_ccache      ccache);
```

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

The **krb5_cc_get_name()** function returns the name of the credentials cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

Return Value

The returned name does not include the credentials cache type prefix. This is a read-only value and must not be freed by the application.

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

---

krb5_cc_get_principal()—Get Principal From a Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_get_principal(
    krb5_context     context,
    krb5_ccache      ccache);
    krb5_principal *     principal);
```

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

The **krb5_cc_get_principal()** function returns the principal associated with the credentials cache. The principal name is set by the **krb5_cc_initialize()** routine. This is the default client principal for tickets stored in the credentials cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

principal (Output)
The principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_cc_get_type()—Get Credentials Cache Type

Syntax

```
#include <krb5.h>

char * krb5_cc_get_type(
    krb5_context     context,
    krb5_ccache      ccache);
```

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

The **krb5_cc_get_type()** function returns the credentials cache type. For example, the string ″FILE″ might be returned.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

Return Value

The value returned is a read-only value and must not be freed by the application.

Error Messages

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

API introduced: V5R1

krb5_cc_initialize()—Initialize Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_initialize(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_principal     principal);
```

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

The **krb5_cc_initialize()** function initializes a credentials cache. Any existing credentials are discarded and the principal name for the cache is set to the value specified. The principal name is the default client name for tickets that will be placed in the cache. A new cache must be initialized before tickets can be stored in the cache.

Authorities

When the credentials cache is of type ″FILE″ (see **krb5_cc_resolve()** for more information on cache types), the default behavior is that the credentials cache file is created in the /QIBM/UserData/OS400/NetworkAuthentication/creds directory. The placement of the credentials cache file can be changed by setting the KRB5CCNAME environment variable.

If the credentials cache file does not reside in the default directory, the following authorities are required:

Object Referred to	Data Authority Required
Each directory in the path name preceding the parent directory	*X
Parent directory if cache file is being created	*WX
Cache file, if being reused	*RW

If the credentials cache file resides in the default directory, the following authorities are required:

Object Referred to	Data Authority Required	Object Authority Required
All directories in the path name	*X	None
Credentials cache file	*RW	None

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

principal (Input)
The default principal for the cache.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_cc_next_cred()—Get Next Entry From a Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_next_cred(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_cc_cursor *     cursor,
    krb5_creds *     creds);
```

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

The **krb5_cc_next_cred()** function reads the next entry from the credentials cache and returns it to the application. The **krb5_cc_start_seq_get()** routine must be called to begin the sequential read operation. The **krb5_cc_next_cred()** routine then is called repeatedly to read cache entries. Finally, the **krb5_cc_end_seq_get()** routine is called when no more entries are to be read.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

cursor (Input/Output)
The cursor created by the **krb5_cc_start_seq_get()** routine. The cursor is updated upon successful completion of this routine.

creds (Output)
The contents of the cache entry. The **krb5_free_cred_contents()** routine should be called to release the credentials contents when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_cc_register()—Define New Credentials Cache Type

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_register(
    krb5_context     context,
    krb5_cc_ops *      ops,
    krb5_boolean     override);
```

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

The **krb5_cc_register()** function allows an application to define a new credentials cache type. Once the new type is registered, it can be used by any thread in the current process and activation group. The type is not known outside the current process and activation group, and is no longer registered when the application ends.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ops (Input)
The credentials cache operations vector. This vector defines the routines that are called to perform the various credentials cache operations for the new cache type.

override (Input)
Whether to override an existing definition for the same type. An error is returned if the type is already registered and **FALSE** is specified for this parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_cc_remove_cred()—Remove Entry

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_remove_cred(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_flags       flags,
    krb5_creds *     mcreds);
```

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

The **krb5_cc_remove_cred()** function removes matching entries from the credentials cache. The client principal must always match. The **KRB5_TC_MATCH_SRV_NAMEONLY** flag controls how much of the server principal must match.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

flags (Input)
The search flags that are used to determine whether a particular cache entry should be removed. The following symbolic definitions are provided for the various flags and should be ORed together to set the desired search flags:

KRB5_TC_MATCH_TIMES (x'00000001')	The *renew_till* and *endtime* values in the cache entry must be greater than the values in the match credentials. A *time* value will be ignored if it is zero.
KRB5_TC_MATCH_IS_SKEY (x'00000002')	The *is_skey* flag in the cache entry must be the same as the *is_skey* flag in the match credentials.
KRB5_TC_MATCH_FLAGS (x'00000004')	All of the flags set in the match credentials must also be set in the cache entry.
KRB5_TC_MATCH_TIMES_EXACT (x'00000008')	The time fields in the cache entry must match exactly the time fields in the match credentials.
KRB5_TC_MATCH_FLAGS_EXACT (x'00000010')	The flags in the cache entry must match exactly the flags in the match credentials.
KRB5_TC_MATCH_AUTHDATA (x'00000020')	The authorization data in the cache entry must be identical to the authorization data in the match credentials.
KRB5_TC_MATCH_SRV_NAMEONLY (x'00000040')	Only the name portion of the server principal in the cache entry needs to match the server principal in the match credentials. The realm values may be different. If this flag is not set, the complete principal name must match.
KRB5_TC_MATCH_2ND_TKT (x'00000080')	The second ticket in the cache entry must match exactly the second ticket in the match credentials.
KRB5_TC_MATCH_KTYPE (x'00000100')	The encryption key type in the cache entry must match the encryption key type in the match credentials.

mcreds (Input)
The match credentials. Fields from these credentials are matched with fields in the cache entries based on the search flags. The client and server principals must always be set in the match credentials, no matter what search flags are specified.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The **krb5_cc_remove_cred()** routine is not supported for the FILE and MEMORY cache types and will return an error code of **KRB5_CC_OP_NOT_SUPPORTED**.

API introduced: V5R1

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

---

krb5_cc_resolve()—Resolve Credentials Cache Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_resolve(
    krb5_context     context,
    char *       cache_name,
    krb5_ccache *      ccache);
```

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

The **krb5_cc_resolve()** function resolves a credentials cache name and returns a handle that can be used to access the cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

cache_name (Input)
The credentials cache name in the format ″type:name″. The type must be a registered credentials cache type and the name must uniquely identify a particular credentials cache of the specified type.

ccache (Output)
The credentials cache handle. The **krb5_cc_close()** or **krb5_cc_destroy()** routine should be called to release the handle when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos runtime supports two credentials cache types: **FILE** and **MEMORY**. Additional credentials cache types can be registered by the application by calling the **krb5_cc_register()** routine. If no type is specified, the default is **FILE**.

API introduced: V5R1

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

---

krb5_cc_retrieve_cred()—Retrieve a Set of Credentials

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_retrieve_cred(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_flags       flags,
    krb5_creds *     mcreds,
    krb5_creds *     creds);
```

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

The **krb5_cc_retrieve_cred()** function searches the credentials cache and returns an entry that matches the credentials specified. The client principal must always match. The **KRB5_TC_MATCH_SRV_NAMEONLY** flag controls how much of the server principal must match.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

flags (Input)
The search flags that are used to determine whether a particular cache entry should be returned to the caller. The following symbolic definitions are provided for the various flags and should be ORed together to set the desired search flags:

KRB5_TC_MATCH_TIMES (x'00000001')	The *renew_till* and *endtime* values in the cache entry must be greater than the values in the match credentials. A *time* value will be ignored if it is zero.
KRB5_TC_MATCH_IS_SKEY (x'00000002')	The *is_skey* flag in the cache entry must be the same as the *is_skey* flag in the match credentials.
KRB5_TC_MATCH_FLAGS (x'00000004')	All of the flags set in the match credentials must also be set in the cache entry.
KRB5_TC_MATCH_TIMES_EXACT (x'00000008')	The time fields in the cache entry must match exactly the time fields in the match credentials.
KRB5_TC_MATCH_FLAGS_EXACT (x'00000010')	The flags in the cache entry must match exactly the flags in the match credentials.
KRB5_TC_MATCH_AUTHDATA (x'00000020')	The authorization data in the cache entry must be identical to the authorization data in the match credentials.
KRB5_TC_MATCH_SRV_NAMEONLY (x'00000040')	Only the name portion of the server principal in the cache entry needs to match the server principal in the match credentials. The realm values may be different. If this flag is not set, the complete principal name must match.
KRB5_TC_MATCH_2ND_TKT (x'00000080')	The second ticket in the cache entry must match exactly the second ticket in the match credentials.
KRB5_TC_MATCH_KTYPE (x'00000100')	The encryption key type in the cache entry must match the encryption key type in the match credentials.
KRB5_TC_SUPPORTED_KTYPES (x'00000200')	The encryption key type in the cache entry must be one of the encryption types specified by the default_tgs_enctypes value in the Kerberos configuration profile. If the default_tgs_enctypes value contains multiple encryption types, the list will be processed from left to right and the first matching credential will be returned.

mcreds (Input)
The match credentials. Fields from these credentials are matched with fields in the cache entries based on the search flags. The client and server principals must always be set in the match credentials, no matter what search flags are specified.

creds (Output)
The contents of the matched cache entry. The **krb5_free_cred_contents()** routine should be called to release the credentials contents when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_cc_set_default_name()—Set Default Credentials Cache Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_set_default_name(
    krb5_context     context,
    const char*          name);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Yes. See "Usage Notes."

The **krb5_cc_set_default_name()** routine sets the name of the default credentials cache for the Kerberos context. Specifying NULL for the name will cause the normal search order to be used to determine the default credentials cache name. Refer to **krb5_cc_default_name()** for a description of the search order.

Authorities

None.

Parameters

context (Input)
The Kerberos context.

name (Input)
The default credentials cache name.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

The **krb5_cc_set_default_name()** routines is not thread-safe unless a separate Kerberos context is used for each thread.

API introduced: V5R2

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

krb5_cc_set_flags()—Set Credentials Cache Processing Flags

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_set_flags(
    krb5_context     context,
    krb5_ccache      ccache);
    krb5_flags       flags);
```

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

The **krb5_cc_set_flags()** function sets the processing flags for the credentials cache. The interpretation of the flags depends on the cache type.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*R

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

flags (Input)
The flags. The allowable flags depends on the cache type. See usage notes.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The **MEMORY** cache type does not support the **krb5_cc_set_flags()** routine and will return **KRB5_CC_OP_NOT_SUPPORTED**.
2. The **FILE** cache type supports the **KRB5_TC_OPENCLOSE** (x'00000001') flag only. If this flag is specified, the credentials cache file is opened each time a credentials cache routine is called and then closed before returning to the caller. This is the default behavior if the **krb5_cc_set_flags()** routine is not called. If this flag is not specified, the credentials cache file is opened and remains open until the credentials cache is closed by the **krb5_cc_close()** or **krb5_cc_destroy()** routine. The sequential read routines are exceptions. Regardless of the **KRB5_TC_OPENCLOSE** flag setting, the credentials cache file is opened when the **krb5_cc_start_seq_get()** routine is called and remains open until the **krb5_cc_end_seq_get()** routine is called.

API introduced: V5R1

krb5_cc_start_seq_get()—Start Sequentially Retrieving Entries from a Credentials Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_start_seq_get(
    krb5_context     context,
    krb5_ccache      cache,
    krb5_cc_cursor *     cursor);
```

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

The **krb5_cc_start_seq_get()** function starts sequentially retrieving entries from the credentials cache. The **krb5_cc_next_cred()** routine is called repeatedly to retrieve each successive cache entry. The **krb5_cc_end_seq_get()** routine is called at the completion of the read operations.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*R

Parameters

context (Input)
The Kerberos context.

cache (Input)
The credentials cache handle.

cursor (Output)
The cursor. The **krb5_cc_end_seq_get()** routine should be called to release the cursor at the completion of the sequential read operations.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The credentials cache is locked when the **krb5_cc_start_seq_get()** routine is called and remains locked until the **krb5_cc_end_seq_get()** routine is called. Write access to the cache by other processes and threads is blocked until the cache is unlocked. After the **krb5_cc_start_seq_get()** routine has been called, the current thread may not call any other credentials cache functions except **krb5_cc_next_cred()** and **krb5_cc_end_seq_get()** for the specified cache.

API introduced: V5R1

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

---

krb5_cc_store_cred()—Store New Set of Credentials

Syntax

```
#include <krb5.h>

krb5_error_code krb5_cc_store_cred(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_creds *     creds);
```

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

The **krb5_cc_store_cred()** function stores a new set of Kerberos credentials in the credentials cache. Existing credentials for the same client/server pair are not removed, even if they have expired. Credentials are stored first-in, first-out, which means that newer credentials are retrieved after older credentials.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache handle.

creds (Input)
The Kerberos credentials.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_change_password()—Change Password

Syntax

```
#include <krb5.h>

krb5_error_code krb5_change_password(
    krb5_context     context,
    krb5_creds *     creds,
    char *                          newpw,
    int *                           result_code,
    krb5_data *                     result_code_string,
    krb5_data *                     result_string);
```

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

The **krb5_change_password()** function changes the password for the principal identified by the supplied credentials. The password change server will apply any applicable password policy checks before changing the password. The password change will be rejected if the policy checks are not successful.

Authorities

None

Parameters

context (Input)
The Kerberos context.

creds (Input)
The credentials for the request. This must be an initial ticket to the kadmin/changepw service for the principal whose password is to be changed.

newpw (Input)
The new password for the principal.

result_code (Output)
Results code for the change password request.

KRB5_KPASSWD_SUCCESS (0)	password changed
KRB5_KPASSWD_MALFORMED (1)	request packet incorrect
KRB5_KPASSWD_HARDERROR (2)	password server error
KRB5_KPASSWD_AUTHERROR (3)	authentication error
KRB5_KPASSWD_SOFTERROR (4)	password changed rejected

result_code_string (Output)

Text description associated with the result code. Specify NULL for this parameter if the text description is not needed. The text description should be released when it is no longer needed by calling the krb5_free_string() function.

result_string (Output)
Additional information provided by the password change server. Specify NULL for this parameter if the additional information is not needed. The result string should be released when it is no longer needed by calling the **krb5_free_string()** function.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned. The password will not have been changed unless both the function return value and the result code are zero.

Error Messages

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

API introduced: V5R2

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

---

krb5_copy_address()—Copy a Kerberos Address to a New Structure

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_address(
    krb5_context   context,
    krb5_const krb5_address *  from_addr,
    krb5_address **    to_addr);
```

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

The **krb5_copy_address()** function copies a Kerberos address to a new structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_addr (Input)
The address to be copied.

to_addr (Output)
The new krb5_address structure. The **krb5_free_address()** routine should be called to release the address when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_copy_addresses()—Copy an Array of Kerberos Addresses

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_addresses(
    krb5_context     context,
    krb5_address * krb5_const *  from_addrs,
    krb5_address ***     to_addrs);
```

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

The **krb5_copy_addresses()** function copies an array of Kerberos address structures.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_addrs (Input)
The array of addresses to be copied. The last array entry must be a NULL pointer.

to_addrs (Output)
The new krb5_address array. The **krb5_free_addresses()** routine should be called to release the address array when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_copy_authdata()—Copy an Array of Authorization Data Structures

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_authdata(
    krb5_context     context,
    krb5_authdata * krb5_const * from_authdata,
    krb5_authdata ***      to_authdata);
```

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

The **krb5_copy_authdata()** function copies an array of authorization data structures.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_authdata (Input)
The array of krb5_authdata structures. The last array entry must be a **NULL** pointer.

to_authdata (Output)
The new array of krb5_authdata structures. The **krb5_free_authdata()** routine should be called to release the array when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_copy_authenticator()—Copy a Kerberos Authenticator

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_authenticator(
    krb5_context     context,
    krb5_const krb5_authenticator *  from_authent,
    krb5_authenticator **    to_authent);
```

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

The **krb5_copy_authenticator()** function copies a Kerberos authenticator.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_authent (Input)
The authenticator to be copied.

to_authent (Output)
The copied authenticator. The **krb5_free_authenticator()** routine should be called to release the authenticator when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_copy_checksum()—Copy a Kerberos Checksum

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_checksum(
    krb5_context     context,
    krb5_const krb5_checksum *   from_cksum,
    krb5_checksum **     to_cksum);
```

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

The **krb5_copy_checksum()** function copies a Kerberos checksum.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_cksum (Input)
The checksum to be copied.

to_cksum (Output)
The copied checksum. The **krb5_free_checksum()** routine should be called to release the checksum when it is no longer needed.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_copy_creds()—Copy Kerberos Credentials

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_creds(
    krb5_context     context,
    krb5_const krb5_creds *    from_creds,
    krb5_creds **      to_creds);
```

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

The **krb5_copy_creds()** function copies Kerberos credentials.

Authorities
No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_creds (Input)
The credentials to be copied.

to_creds (Output)
The copied credentials. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_copy_data()—Copy a Kerberos Data Object

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_data(
    krb5_context     context,
    krb5_const krb5_data *     from_data,
    krb5_data **     to_data);
```

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

The **krb5_copy_data()** function copies a Kerberos data object that is represented by a krb5_data structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_data (Input)
The data object to be copied.

to_data (Output)
The copied data object. The **krb5_free_data()** routine should be called to release the data object when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_copy_keyblock()—Copy a Kerberos Keyblock

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_keyblock(
    krb5_context     context,
    krb5_const krb5_keyblock *   from_keyblock,
    krb5_keyblock **     to_keyblock);
```

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

The **krb5_copy_keyblock()** function copies a Kerberos keyblock.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_keyblock (Input)
The keyblock to be copied.

to_keyblock (Output)
The copied keyblock. The **krb5_free_keyblock()** routine should be called to release the keyblock when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_copy_keyblock_contents()—Copy Contents of a Kerberos Keyblock

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_keyblock_contents(
    krb5_context     context,
    krb5_const krb5_keyblock *   from_keyblock,
    krb5_keyblock *      to_keyblock);
```

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

The **krb5_copy_keyblock_contents()** function copies the contents of a Kerberos keyblock into an existing keyblock. The contents of the output keyblock are not released before performing the copy.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_keyblock (Input)
The keyblock to be copied.

to_keyblock (Output)
The contents of the input keyblock. The **krb5_free_keyblock_contents()** routine should be called to release the contents of the keyblock when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_copy_principal()—Copy a Kerberos Principal

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_principal(
    krb5_context     context,
    krb5_const_principal     from_princ,
    krb5_principal *     to_princ);
```

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

The **krb5_copy_principal()** function copies a Kerberos principal.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_princ (Input)
The principal to be copied.

to_princ (Output)
The copied principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_copy_ticket()—Copy a Kerberos Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_copy_ticket(
    krb5_context     context,
    krb5_const krb5_ticket *     from_ticket,
    krb5_ticket **     to_ticket);
```

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

The **krb5_copy_ticket()** function copies a Kerberos ticket.

Authorities
No authorities are required.

Parameters

context (Input)
The Kerberos context.

from_ticket (Input)
The ticket to be copied.

to_ticket (Output)
The copied ticket. The **krb5_free_ticket()** routine should be called to release the ticket when it is no longer needed.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_free_address()—Free Storage Assigned to a Kerberos Address

Syntax

```
#include <krb5.h>

void krb5_free_address(
    krb5_context     context,
    krb5_address *     addr);
```

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

The **krb5_free_address()** function releases the storage assigned to the contents of a krb5_address structure and then releases the krb5_address structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

addr (Input)
The krb5_address to be released.

Return Value

This routine does not return a value.

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

krb5_free_addresses()—Free Storage Assigned to Array of Kerberos Addresses

Syntax

```
#include <krb5.h>

void krb5_free_addresses(
    krb5_context     context,
    krb5_address **      addrs);
```

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

The **krb5_free_addresses()** function releases the storage assigned to an array of krb5_address structures. Each krb5_address structure is released and then the pointer array itself is released.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

addrs (Input)
The array of addresses to be released. The last entry in the array must be a **NULL** pointer.

Return Value

This routine does not return a value.

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

---

krb5_free_ap_rep_enc_part()—Free Storage Assigned to AP_REP Message Encrypted Part

Syntax

```
#include <krb5.h>

void krb5_free_ap_rep_enc_part(
    krb5_context     context,
    krb5_ap_rep_enc_part   enc_part);
```

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

The **krb5_free_ap_rep_enc_part()** function releases the storage assigned to the decrypted portion of an AP_REP message.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

enc_part (Input)
The reply to be released.

Return Value
This routine does not return a value.

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

---

krb5_free_authdata()—Free Storage Assigned to Array of Authentication Data

Syntax

```
#include <krb5.h>

void krb5_free_authdata(
    krb5_context     context,
    krb5_authdata **     authdata);
```

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

The **krb5_free_authdata()** function releases the storage assigned to an array of krb5_authdata structures. Each krb5_authdata structure is released and then the pointer array itself is released.

Authorities
No authorities are required.

Parameters

context (Input)
The Kerberos context.

authdata (Input)
The array to be released. The last entry in the array must be a **NULL** pointer.

Return Value
This routine does not return a value.

Error Messages

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

API introduced: V5R1

krb5_free_authenticator()—Free Storage Assigned to Authenticator

Syntax

```
#include <krb5.h>

void krb5_free_authenticator(
    krb5_context     context,
    krb5_authenticator *   authent);
```

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

The **krb5_free_authenticator()** function releases the storage assigned to the contents of a krb5_authenticator structure and then releases the krb5_authenticator structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

authent (Input)
The krb5_authenticator to be released.

Return Value

This routine does not return a value.

Error Messages

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

API introduced: V5R1

krb5_free_authenticator_contents()—Free Storage Assigned to Contents of Authenticator

Syntax

```
#include <krb5.h>

void krb5_free_authenticator_contents(
    krb5_context     context,
    krb5_authenticator *   authent);
```

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

The **krb5_free_authenticator_contents()** function releases the storage assigned to the contents of a krb5_authenticator structure. Unlike the **krb5_free_authenticator()** function, the **krb5_free_authenticator_contents()** function does not free the krb5_authenticator structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

authent (Input)
The krb5_authenticator to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_checksum()—Free Storage Assigned to Checksum

Syntax

```
#include <krb5.h>

void krb5_free_checksum(
    krb5_context     context,
    krb5_checksum *      cksum);
```

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

The **krb5_free_checksum()** function releases the storage assigned to a krb5_checksum structure and then releases the krb5_checksum structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

cksum (Input)
The krb5_checksum to be released.

Return Value
This routine does not return a value.

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

---

krb5_free_cksumtypes()—Free Checksum Types

Syntax

```
#include <krb5.h>

krb5_error_code krb5_free_cksumtypes(
    krb5_context       context,
    krb5_cksumtype                    cksumtypes);
```

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

The **krb5_free_cksumtypes()** function releases storage assigned to an array of checksum types.

Authorities
None.

Parameters

context (Input)
The Kerberos context.

cksumtypes (Input)
The array of checksum types to be released.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R2

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

krb5_free_context()—Free Kerberos Context

Syntax

```
#include <krb5.h>

void krb5_free_context(
    krb5_context     context);
```

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

The **krb5_free_context()** function releases a context that was created by the **krb5_init_context()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The context to be released.

Return Value

This routine does not return a value.

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

krb5_free_creds()—Free Storage Assigned to a Credential

Syntax

```
#include <krb5.h>

void krb5_free_creds(
    krb5_context     context,
    krb5_creds *     creds);
```

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

The **krb5_free_creds()** function releases the storage assigned to the contents of a krb5_creds structure and then releases the krb5_creds structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

creds (Input)
The credential.

Return Value

This routine does not return a value.

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

---

krb5_free_cred_contents()—Free Storage Assigned to Contents of a Credential

Syntax

```
#include <krb5.h>

void krb5_free_cred_contents(
    krb5_context     context,
    krb5_creds *     creds);
```

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

The **krb5_free_cred_contents()** function releases the storage assigned to the contents of a krb5_creds structure. Unlike the **krb5_free_creds()** routine, the **krb5_free_cred_contents()** routine does not release the krb5_creds structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

creds (Input)
The credential containing the contents to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_data()—Free Storage Assigned to a Kerberos Data Object

Syntax

```
#include <krb5.h>

void krb5_free_data(
    krb5_context     context,
    krb5_data *      data);
```

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

The **krb5_free_data()** function releases the storage assigned to a Kerberos data object represented by a krb5_data structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

data (Input)
The data object.

Return Value

This routine does not return a value.

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

---

krb5_free_data_contents()—Free Storage Assigned to Contents of a Kerberos Data Object

Syntax

```
#include <krb5.h>

void krb5_free_data_contents(
    krb5_context     context,
    krb5_data *      data);
```

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

The **krb5_free_data_contents()** function releases the storage assigned to the contents of a Kerberos data object represented by a krb5_data structure. Unlike the **krb5_free_data()** routine, the **krb5_free_data_contents()** routine does not release the krb5_data structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

data (Input)
The data object.

Return Value

This routine does not return a value.

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

---

krb5_free_enctypes()—Free Storage Assigned to Array of Encryption Types

Syntax

```
#include <krb5.h>

void krb5_free_enctypes(
    krb5_context     context,
    krb5_enctype *     enctypes);
```

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

The **krb5_free_enctypes()** function releases the storage assigned to an array of encryption types.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

enctypes (Input)
The array of encrytion types to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_enc_tkt_part()—Free Storage Assigned to Encrypted Ticket Part

Syntax

```
#include <krb5.h>

void krb5_free_enc_tkt_part(
    krb5_context     context,
    krb5_enc_tkt_part *    enc_tkt);
```

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

The **krb5_free_enc_tkt_part()** function releases the storage assigned to to the krb5_enc_tkt_part structure and then releases the krb5_enc_tkt_part structure itself. The krb5_enc_tkt_part structure is created when a ticket is decrypted and decoded.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

enc_tkt (Input)
The krb5_enc_tkt_part structure to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_error()—Free Storage Assigned to Kerberos Error Message

Syntax

```
#include <krb5.h>

void krb5_free_error(
    krb5_context     context,
    krb5_error *     error);
```

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

The **krb5_free_error()** function releases the storage assigned to the krb5_error structure and then releases the krb5_error structure itself. The krb5_error structure is created when a Kerberos error message is processed by the **krb5_rd_error()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

error (Input)
The krb5_error structure to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_host_realm()—Free Storage Assigned to Realm List

Syntax

```
#include <krb5.h>

krb5_error_code krb5_free_host_realm(
    krb5_context     context,
    char * krb5_const *    realm_list);
```

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

The **krb5_free_host_realm()** function releases the storage assigned to a realm list.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

realm_list (Input)
The realm list to be released.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_free_kdc_rep()—Free Storage Assigned to KDC Reply

Syntax

```
#include <krb5.h>

void krb5_free_kdc_rep(
    krb5_context     context,
    krb5_kdc_rep *     reply);
```

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

The **krb5_free_kdc_rep()** function releases the contents of the krb5_kdc_rep structure and then releases the krb5_kdc_rep structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

reply (Input)
The KDC reply to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_keyblock()—Free Storage Assigned to a Keyblock

Syntax

```
#include <krb5.h>

void krb5_free_keyblock(
    krb5_context     context,
    krb5_keyblock *      keyblock);
```

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

The **krb5_free_keyblock()** function releases the contents of the krb5_keyblock structure and then releases the krb5_keyblock structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

keyblock (Input)
The keyblock to be released.

Return Value

This routine does not return a value.

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

krb5_free_keyblock_contents()—Free Storage Assigned to Contents of a Keyblock

Syntax

```
#include <krb5.h>

void krb5_free_keyblock_contents(
    krb5_context     context,
    krb5_keyblock *      keyblock);
```

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

The **krb5_free_keyblock_contents()** function releases the contents of the krb5_keyblock structure. Unlike the **krb5_free_keyblock()** routine, the **krb5_free_keyblock_contents()** routine does not release the krb5_keyblock structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

keyblock (Input)
The keyblock that contains the contents to be released.

Return Value

This routine does not return a value.

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

krb5_free_krbhst()—Free Storage Assigned to Host List

Syntax

```
#include <krb5.h>

krb5_error_code krb5_free_krbhst(
    krb5_context     context,
    char * krb5_const *    host_list);
```

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

The **krb5_free_krbhst()** function releases the storage assigned to a host list.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

host_list (Input)
The host list to be released.

Return Value

The function return value is 0 if no errors occur. Otherwise, it is a Kerberos error code.

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

---

krb5_free_principal()—Free Storage Assigned to Principal

Syntax

```
#include <krb5.h>

void krb5_free_principal(
    krb5_context     context,
    krb5_principal     principal);
```

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

The **krb5_free_principal()** function releases the storage assigned to a krb_5 principal.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

principal (Input)
The krb5_principal to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_string()—Free Storage Assigned to Character String

Syntax

```
#include <krb5.h>

void krb5_free_string(
    krb5_context     context,
    char *       string);
```

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

The **krb5_free_string()** function releases the storage assigned to a character string.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

string (Input)
The character string to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_tgt_creds()—Free Storage Assigned to Array of Credentials

Syntax

```
#include <krb5.h>

void krb5_free_tgt_creds(
    krb5_context     context,
    krb5_creds **      creds);
```

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

The **krb5_free_tgt_creds()** function releases the storage assigned to an array of krb5_creds structures. Each krb5_creds structure is released and then the pointer array itself is released.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

creds (Input)
The credentials array to be released. The last entry in the array must be a **NULL** pointer.

Return Value

This routine does not return a value.

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

---

krb5_free_ticket()—Free Storage Assigned to a Ticket

Syntax

```
#include <krb5.h>

void krb5_free_ticket(
    krb5_context     context,
    krb5_ticket *      ticket);
```

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

The **krb5_free_ticket()** function releases the storage assigned to a krb5_ticket structure and then releases the krb5_ticket structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ticket (Input)
The krb5_ticket to be released.

Return Value

This routine does not return a value.

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

---

krb5_free_tickets()—Free Storage Assigned to Array of Tickets

Syntax

```
#include <krb5.h>

void krb5_free_tickets(
    krb5_context     context,
    krb5_ticket **     tickets);
```

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

The **krb5_free_tickets()** function releases the storage assigned to an array of krb5_ticket structures. Each krb5_ticket structure is released and then the pointer array itself is released.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

tickets (Input)
The array to be released. The last entry in the array must be a **NULL** pointer.

Return Value

This routine does not return a value.

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

krb5_generate_seq_number()—Generate Random Sequence Number

Syntax

```
#include <krb5.h>

krb5_error_code krb5_generate_seq_number(
    krb5_context     context,
    krb5_const krb5_keyblock *   key,
    krb5_int32 *     seqno);
```

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

The **krb5_generate_seq_number()** function generates a random sequence number based on the supplied key.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

key (Input)
The key used to generate the random sequence number.

seqno (Output)
The random sequence number.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_generate_subkey()—Generate Subsession Key

Syntax

```
#include <krb5.h>

krb5_error_code krb5_generate_subkey(
    krb5_context     context,
    krb5_const krb5_keyblock *   key,
    krb5_keyblock **     subkey);
```

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

The **krb5_generate_subkey()** function generates a random subsession key that is based on the supplied session key.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

key (Input)
The session key.

subkey (Output)
The generated subsession key. The **krb5_free_keyblock()** routine should be called to release the key when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_gen_replay_name()—Generate Replay Cache Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_gen_replay_name(
    krb5_context     context,
    krb5_const krb5_address *    inaddr,
    krb5_const char *      unique,
    char **        string);
```

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

The **krb5_gen_replay_name()** function generates a unique replay cache name based on the Kerberos address supplied by the caller. The *unique* parameter is used to differentiate this replay cache from others currently in use on the system. The generated cache name consists of the unique portion concatenated with the hexadecimal representation of the Kerberos address.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

inaddr (Input)
The address to be incorporated into the cache name.

unique (Input)
The unique portion of the replay cache name.

string (Output)
The generated replay cache name. The **krb5_free_string()** function should be called to free the string when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_credentials()—Get Service Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_credentials(
    krb5_context     context,
    krb5_const krb5_flags    options,
    krb5_ccache      ccache,
    krb5_creds *       in_cred,
    krb5_creds **      out_cred);
```

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

The **krb5_get_credentials()** function obtains a service ticket for the requested server. This routine is the normal way for an application to obtain a service ticket. If the service ticket is already in the credentials cache, the **krb5_get_credentials()** routine returns the cached ticket. Otherwise, the **krb5_get_credentials()** routine calls the **krb5_get_cred_from_kdc()** routine to obtain a service ticket from the Kerberos server.

The **krb5_get_credentials()** routine stores any tickets obtained during its processing in the credentials cache. This includes the requested service ticket, as well as any ticket-granting tickets required to obtain the service ticket.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

options (Input)
The option flags as follows:

KRB5_GC_USER_USER (x'00000001')	Obtain a user-to-user ticket.
KRB5_GC_CACHED (x'00000002')	Do not obtain a service ticket if one is not found in the credentials cache.

ccache (Input)
The credentials cache to be used. The initial ticket-granting ticket must already be in the cache.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time. The key encryption type can be set to override the default ticket encryption type.

out_cred (Output)
The service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. If **KRB5_GC_CACHED** is specified, the **krb5_get_credentials()** routine searches only the credentials cache for a service ticket.
2. If **KRB5_GC_USER_USER** is specified, the **krb5_get_credentials()** routine gets credentials for user-to-user authentication. In user-to-user authentication, the secret key for the server is the session key from the server's ticket-granting ticket. The ticket-granting ticket is passed from the server to the client over the network. (This is safe since the ticket-granting ticket is encrypted in a key known only

by the Kerberos server.) The client must then pass this ticket-granting ticket to **krb5_get_credentials()** as the second ticket in the request credentials. The Kerberos server uses this ticket-granting ticket to construct a user-to-user ticket that can be verified by the server using the session key from its ticket-granting ticket.

API introduced: V5R1

---

krb5_get_credentials_renew()—Renew Service Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_credentials_renew(
    krb5_context     context,
    krb5_const krb5_flags    options,
    krb5_ccache      ccache,
    krb5_creds *       in_cred,
    krb5_creds **      out_cred);
```

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

The **krb5_get_credentials_renew()** function renews a service ticket for the rquested service. Upon successful completion, the credentials cache is reinitialized and the service ticket is stored in the cache.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

options (Input)
The option flags as follows:

KRB5_GC_USER_USER (x'00000001') Obtain a user-to-user ticket.

ccache (Input)
The credentials cache to be used.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time.

out_cred (Output)
The service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_credentials_validate()—Validate Service Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_credentials_validate(
    krb5_context     context,
    krb5_const krb5_flags    options,
    krb5_ccache      ccache,
    krb5_creds *       in_cred,
    krb5_creds **      out_cred);
```

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

The **krb5_get_credentials_validate()** routine validates a service ticket for the requested service. Upon successful completion, the credentials cache is reinitialized and the service ticket is stored in the cache.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

options (Input)
The option flags as follows:

KRB5_GC_USER_USER (x'00000001') Obtain a user-to-user ticket.

ccache (Input)
The credentials cache to be used.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time.

out_cred (Output)
The service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_cred_from_kdc()—Get Service Ticket from Kerberos KDC Server

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_cred_from_kdc(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_creds *     in_cred,
    krb5_creds **      out_cred,
    krb5_creds ***     tgts);
```

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

The **krb5_get_cred_from_kdc()** function obtains a service ticket from the Kerberos Key Distribution Center (KDC) server. The credentials are not stored in the credentials cache. (The application should store them in the cache if appropriate.) The application should not call **krb5_get_cred_from_kdc()** if the requested service ticket is already in the credentials cache.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache. The initial ticket-granting ticket for the local realm must already be in the cache. The Kerberos runtime obtains additional ticket-granting tickets as needed if the target server is not in the local realm.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time.

out_cred (Output)
The service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

tgts (Output)
Any new ticket-granting tickets that were obtained while getting the service target from the KDC in the target realm. There may be ticket-granting tickets returned for this parameter even if the Kerberos runtime ultimately was unable to obtain a service ticket from the target KDC. The **krb5_free_tgt_creds()** routine should be called to release the ticket-granting ticket array when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The **krb5_get_cred_from_kdc()** routine obtains any necessary ticket-granting tickets for intermediate realms between the client realm and the server realm. It then calls the **krb5_get_cred_via_tkt()** routine to obtain the actual service ticket. The KDC options are the same as the ticket-granting ticket options. The **KDC_OPT_ENC_TKT_IN_SKEY** (x'00000008') flag is set if the *in_cred* parameter provided a second ticket.

API introduced: V5R1

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

---

krb5_get_cred_from_kdc_renew()—Renew Service Ticket Obtained from Kerberos KDC Server

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_cred_from_kdc_renew(
    krb5_context     context,
```

```
krb5_ccache      ccache,
krb5_creds *     in_cred,
krb5_creds **      out_cred,
krb5_creds ***     tgts);
```

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

The **krb5_get_cred_from_kdc_renew()** function renews a service ticket obtained from the Kerberos Key Distribution Center (KDC) server. The credentials are not stored in the credentials cache. (The application should store them in the cache if appropriate.)

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache. The initial ticket-granting ticket for the local realm must already be in the cache. The Kerberos runtime obtains additional ticket-granting tickets as needed if the target server is not in the local realm.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time.

out_cred (Output)
The renewed service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

tgts (Output)
Any new ticket-granting tickets that were obtained while getting the service target from the KDC in the target realm. There may be ticket-granting tickets returned for this parameter even if the Kerberos runtime ultimately was unable to obtain a service ticket from the target KDC. The **krb5_free_tgt_creds()** routine should be called to release the ticket-granting ticket array when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The application should call **krb5_get_cred_from_kdc_renew()** to renew a renewable ticket before the ticket end time is reached. A renewable ticket may not be renewed after its end time, even if its *renew_till* time has not been reached yet.

API introduced: V5R1

---

krb5_get_cred_from_kdc_validate()—Validate Service Ticket Obtained from Kerberos KDC Server

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_cred_from_kdc_validate(
    krb5_context     context,
    krb5_ccache      ccache,
    krb5_creds *     in_cred,
    krb5_creds **      out_cred,
    krb5_creds ***     tgts);
```

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

The **krb5_get_cred_from_kdc_validate()** function validates a service ticket obtained from the Kerberos Key Distribution Center (KDC) server. The credentials are not stored in the credentials cache. (The application should store them in the cache if appropriate.)

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

ccache (Input)
The credentials cache. The initial ticket-granting ticket for the local realm must already be in the cache. The Kerberos runtime obtains additional ticket-granting tickets as needed if the target server is not in the local realm.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time.

out_cred (Output)
The validated service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

tgts (Output)
Any new ticket-granting tickets that were obtained while getting the service target from the KDC in the target realm. There may be ticket-granting tickets returned for this parameter even if the Kerberos runtime ultimately was unable to obtain a service ticket from the target KDC. The **krb5_free_tgt_creds()** routine should be called to release the ticket-granting ticket array when it is no longer needed.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes
1. The application should call **krb5_get_cred_from_kdc_validate()** to validate a postdated ticket once the ticket start time has been reached.

API introduced: V5R1

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

---

krb5_get_cred_via_tkt()—Get Service Ticket from Kerberos KDC Server Using Supplied Ticket-granting Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_cred_via_tkt(
    krb5_context     context,
    krb5_creds *     tkt,
    krb5_const krb5_flags    kdc_options,
    krb5_address **      address,
    krb5_creds *       in_cred,
    krb5_creds **      out_cred);
```

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

The **krb5_get_cred_via_tkt()** function obtains a service ticket from the Kerberos Key Distribution Center (KDC) server.

Authorities
No authorities are required.

Parameters

context (Input)
The Kerberos context.

tkt (Input)
The ticket-granting ticket for the realm containing the target server for the service ticket. The client in the ticket-granting ticket must be the same as the client in the request credentials.

kdc_options (Input)
KDC options for the service ticket as follows:

KDC_OPT_FORWARDABLE (x'40000000')	Obtain a forwardable ticket.
KDC_OPT_PROXIABLE (x'10000000')	Obtain a proxiable ticket.
KDC_OPT_ALLOW_POSTDATE (x'04000000')	Allow postdated tickets.
KDC_OPT_RENEWABLE (x'00800000')	Obtain a renewable ticket. The *renew_till* time must be set in the request.
KDC_OPT_RENEWABLE_OK (x'00000010')	A renewable ticket is acceptable if the KDC policy does not allow a ticket to be generated with the requested endtime.
KDC_OPT_ENC_TKT_IN_SKEY (x'00000008')	Encrypt the service ticket in the session key of the second ticket.

address (Input)
The addresses to be placed in the ticket. The ticket addresses determine which host systems can generate requests to use the ticket.

in_cred (Input)
The request credentials. The *client* and *server* fields must be set to the desired values for the service ticket. The *second_ticket* field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time.

out_cred (Output)
The service ticket. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. If the request is for a ticket-granting ticket in a foreign realm, the KDC may return a ticket-granting ticket for an intermediate realm if it is unable to return a ticket-granting ticket for the requested realm. The application should check the server name in the returned ticket-granting ticket. If the ticket-granting ticket is not for the desired realm, the application should call **krb5_get_cred_via_tkt()** again to send the request to the KDC for the realm in the returned ticket-granting ticket and should provide the ticket-granting ticket as the credentials for the request.

API introduced: V5R1

krb5_get_default_in_tkt_ktypes()—Get Default Encryption Types to be Used for Initial Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_default_in_tkt_ktypes(
    krb5_context     context,
    krb5_enctype **      ktypes);
```

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

The **krb5_get_default_in_tkt_ktypes()** function returns the default encryption types that are used when requesting an initial ticket from the Kerberos server. The values are set by the **krb5_set_default_in_tkt_ktypes()** routine or are obtained from the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktypes (Output)
An array of encryption types. The last entry in the array is **ENCTYPE_NULL**. The **krb5_free_enctypes()** routine should be called to release the array of encryption types when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_get_default_realm()—Get Default Realm

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_default_realm(
    krb5_context     context,
    char **        realm);
```

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

The **krb5_get_default_realm()** function returns the default realm for the local system. The default realm is set by the **krb5_set_default_realm()** routine. If the default realm has not been set, it is obtained from the *default_realm* entry in the *[libdefaults]* section of the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

realm (Output)
The realm name. The **krb5_free_string()** routine should be called to free the string when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_default_tgs_ktypes()—Get Default Encryption Types to be Used for Service Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_default_tgs_ktypes(
    krb5_context     context,
    krb5_enctype **      ktypes);
```

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

The **krb5_get_default_tgs_ktypes()** function returns the default encryption types that are used when requesting a service ticket from the Kerberos server. The values are set by the **krb5_set_default_tgs_ktypes()** routine or are obtained from the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktypes (Output)
An array of encryption types. The last entry in the array is **ENCTYPE_NULL**. The **krb5_free_enctypes()** routine should be called to release the array of encryption types when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_host_realm()—Get Kerberos Realm Name for Host Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_host_realm(
    krb5_context     context,
    krb5_const char *      host,
    char ***       realm_list);
```

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

The **krb5_get_host_realm()** function returns a list of Kerberos realm names for the specified host name. The entries in the *[domain_realm]* section of the Kerberos configuration file are used. A direct match takes precedence over a suffix match. The current implementation of this routine returns a single realm name. If no realm name is found, the uppercased host domain is returned as the realm name.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

host (Input)
The host name. The local host name is used if **NULL** is specified for this parameter.

realm (Output)
An array of realm names. The last entry in the array will be a **NULL** pointer. The **krb5_free_host_realm()** routine should be called to release the realm list when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_in_tkt_with_keytab()—Get Initial Ticket Using Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_in_tkt_with_keytab(
    krb5_context     context,
    krb5_const krb5_flags    options,
    krb5_address * krb5_const *  addrs,
    krb5_enctype *     enctype,
    krb5_preauthtype *     pre_auth_types,
    krb5_const krb5_keytab   keytab,
    krb5_ccache      ccache,
    krb5_creds *     creds,
    krb5_kdc_rep **      ret_as_reply);
```

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

The **krb5_get_in_tkt_with_keytab()** function obtains an initial ticket-granting ticket from the Kerberos Key Distribution Center (KDC) server using a key table. This initial ticket can then be used to obtain service tickets. The client must be in the same realm as the KDC to be able to obtain an initial ticket from the KDC. The initial ticket can be used to obtain tickets in the same realm or in different realms as long as the proper inter-realm trust relationships have been established.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the key table file, if key parameter is NULL	*X
Key table file, if key parameter is NULL	*R
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

options (Input)
The KDC options as follows:

KDC_OPT_FORWARDABLE (x'40000000')	Obtain a forwardable ticket.
KDC_OPT_PROXIABLE (x'10000000')	Obtain a proxiable ticket.
KDC_OPT_ALLOW_POSTDATE (x'04000000')	Allow postdated tickets.
KDC_OPT_RENEWABLE (x'00800000')	Obtain a renewable ticket. The *renew_till* time must be set in the request.
KDC_OPT_RENEWABLE_OK (x'00000010')	A renewable ticket is acceptable if the KDC policy does not allow a ticket to be generated with the requested endtime.

addrs (Input)
The addresses to be placed in the ticket. If **NULL** is specified for this parameter, the local system addresses are used. The address list is an array of krb5_address pointers. The end of the array is indicated by a **NULL** pointer. No addresses are included in the initial ticket if the address array consists of a single **NULL** entry. The ticket addresses determine which host systems can generate requests that use the ticket.

enctype (Input)
An array of encryption types to be used. The last entry in the array must be **ENCTYPE_NULL** (x'00000000'). If **NULL** is specified for this parameter, the default encryption types are used. The following encryption types may be specified:

ENCTYPE_DES_CBC_CRC (x'00000001')	32-bit CRC checksum with DES encryption. This encryption type should be used for interoperability with older levels of Kerberos Version 5.
ENCTYPE_DES_CBC_MD5 (x'00000003')	MD5 checksum with DES encryption.

pre_auth_types (Input)
An array of preauthentication types to be used. The last entry in the array must be **KRB5_PADATA_NONE** (x'00000000'). If **NULL** is specified for this parameter, no preauthentication is done unless required by KDC policy. If multiple preauthentication types are specified, the KDC is supposed to accept the request as long as it recognizes at least one of the preauthentication types. Early implementations of the KDC did not follow this rule and will fail the request if the first preauthentication type is not recognized. The following preauthentication type may be specified:

KRB5_PADATA_ENC_TIMESTAMP (x'00000002')	Encrypted timestamp preauthentication.

keytab (Input)
The key table containing the key for the client principal. The entry with the highest key version number is used. The default key table is used if **NULL** is specified for this parameter.

ccache (Input)
The credentials cache handle. The initial ticket ise stored in the credentials cache for later use by the application. The credentials are not stored if **NULL** is specified for this parameter.

creds (Input/Output)
The credentials that are used to obtain the initial ticket. The *client* and *server* fields must be set.

The *endtime* field may be set to explicitly specify the ticket lifetime or it may be set to zero to use the default ticket lifetime. The *renew_till* field must be set if a renewable ticket is being requested. The *starttime* field must be set if a postdated ticket is being requested.

Upon completion of the request, **creds** are updated with the initial ticket, the session key, and the client address list. The **krb5_free_cred_contents()** or **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

ret_as_reply (Output)
The KDC reply. Specify **NULL** for this parameter if the KDC reply is not needed. The **krb5_free_kdc_rep()** routine should be called to release the reply when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

Top | "Network Authentication Service APIs," on page 1
Security APIs | UNIX-Type APIs
APIs by category

---

krb5_get_in_tkt_with_password()—Get Initial Ticket Using Text Password

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_in_tkt_with_password(
    krb5_context     context,
    krb5_const krb5_flags    options,
    krb5_address * krb5_const *  addrs,
    krb5_enctype *     enctypes,
    krb5_preauthtype *     pre_auth_types,
    krb5_const char *      password,
    krb5_ccache      ccache,
    krb5_creds *     creds,
    krb5_kdc_rep **      ret_as_reply);
```

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

The **krb5_get_in_tkt_with_password()** function obtains an initial ticket-granting ticket from the Kerberos Key Distribution Center (KDC) server using a text password. This initial ticket can then be used to obtain service tickets. The client must be in the same realm as the KDC to be able to obtain an initial ticket from the KDC. The initial ticket can be used to obtain tickets in the same realm or in different realms as long as the proper inter-realm trust relationships have been established.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

options (Input)
The KDC options as follows:

KDC_OPT_FORWARDABLE (x'40000000')	Obtain a forwardable ticket.
KDC_OPT_PROXIABLE (x'10000000')	Obtain a proxiable ticket.
KDC_OPT_ALLOW_POSTDATE (x'04000000')	Allow postdated tickets.
KDC_OPT_RENEWABLE (x'00800000')	Obtain a renewable ticket. The *renew_till* time must be set in the request.
KDC_OPT_RENEWABLE_OK (x'00000010')	A renewable ticket is acceptable if the KDC policy does not allow a ticket to be generated with the requested endtime.

addrs (Input)
The addresses to be placed in the ticket. If **NULL** is specified for this parameter, the local system addresses are used. The address list is an array of krb5_address pointers. The end of the array is indicated by a **NULL** pointer. No addresses are included in the initial ticket if the address array consists of a single **NULL** entry. The ticket addresses determine which host systems can generate requests that use the ticket.

enctypes (Input)
An array of encryption types to be used. The last entry in the array must be **ENCTYPE_NULL** (x'00000000'). If **NULL** is specified for this parameter, the default encryption types are used. The following encryption types may be specified:

ENCTYPE_DES_CBC_CRC (x'00000001')	32-bit CRC checksum with DES encryption. This encryption type should be used for interoperability with older levels of Kerberos Version 5.
ENCTYPE_DES_CBC_MD5 (x'00000003')	MD5 checksum with DES encryption.

pre_auth_types (Input)
An array of preauthentication types to be used. The last entry in the array must be **KRB5_PADATA_NONE** (x'00000000'). If **NULL** is specified for this parameter, no preauthentication is done unless required by KDC policy. If multiple preauthentication types are specified, the KDC is supposed to accept the request as long as it recognizes at least one of the preauthentication types. Early implementations of the KDC did not follow this rule and will fail the request if the first preauthentication type is not recognized. The following preauthentication type may be specified:

KRB5_PADATA_ENC_TIMESTAMP (x'00000002') Encrypted timestamp preauthentication. This preauthentication type should be used for interoperability with a Kerberos KDC.

password (Input)
The password string. This string is converted to a Kerberos key value using the rules for the first encryption type specified by the *enctypes* parameter. The user is prompted to enter the password if **NULL** is specified for this parameter.

ccache (Input)
The credentials cache handle. The initial ticket is stored in the credentials cache for later use by the application. The credentials are not stored if **NULL** is specified for this parameter.

creds (Input/Output)
The credentials that are used to obtain the initial ticket. The *client* and *server* fields must be set. The *endtime* field may be set to explicitly specify the ticket lifetime or it may be set to zero to use the default ticket lifetime. The *renew_till* field must be set if a renewable ticket is being requested. The *starttime* field must be set if a postdated ticket is being requested.

Upon completion of the request, **creds** are updated with the initial ticket, the session key, and the client address list. The **krb5_free_cred_contents()** or **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

ret_as_reply (Output)
The KDC reply. Specify **NULL** for this parameter if the KDC reply is not needed. The **krb5_free_kdc_rep()** routine should be called to release the reply when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_in_tkt_with_skey()—Get Initial Ticket Using Session Key

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_in_tkt_with_skey(
    krb5_context     context,
    krb5_const krb5_flags    options,
    krb5_address * krb5_const *  addrs,
    krb5_enctype *     enctypes,
    krb5_preauthtype *     pre_auth_types,
    krb5_const krb5_keyblock *   key,
    krb5_ccache      ccache,
    krb5_creds *     creds,
    krb5_kdc_rep **      ret_as_reply);
```

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

The **krb5_get_in_tkt_with_skey()** function obtains an initial ticket-granting ticket from the Kerberos Key Distribution Center (KDC) server using a session key. This initial ticket can then be used to obtain service tickets. The client must be in the same realm as the KDC to be able to obtain an initial ticket from the KDC. The initial ticket can be used to obtain tickets in the same realm or in different realms as long as the proper inter-realm trust relationships have been established.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the key table file, if key parameter is NULL	*X
Key table file, if key parameter is NULL	*R
Each directory in the path name preceding the credentials cache file	*X
Credentials cache file	*RW

Parameters

context (Input)
The Kerberos context.

options (Input)
The KDC options as follows:

KDC_OPT_FORWARDABLE (x'40000000')	Obtain a forwardable ticket.
KDC_OPT_PROXIABLE (x'10000000')	Obtain a proxiable ticket.
KDC_OPT_ALLOW_POSTDATE (x'04000000')	Allow postdated tickets.
KDC_OPT_RENEWABLE (x'00800000')	Obtain a renewable ticket. The *renew_till* time must be set in the request.
KDC_OPT_RENEWABLE_OK (x'00000010')	A renewable ticket is acceptable if the KDC policy does not allow a ticket to be generated with the requested endtime.

addrs (Input)
The addresses to be placed in the ticket. If **NULL** is specified for this parameter, the local system addresses are used. The address list is an array of krb5_address pointers. The end of the array is indicated by a **NULL** pointer. No addresses are included in the initial ticket if the address array consists of a single **NULL** entry. The ticket addresses determine which host systems can generate requests that use the ticket.

enctypes (Input)
An array of encryption types to be used. The last entry in the array must be **ENCTYPE_NULL** (x'00000000'). If **NULL** is specified for this parameter, the default encryption types are used. The following encryption types may be specified:

ENCTYPE_DES_CBC_CRC (x'00000001')	32-bit CRC checksum with DES encryption. This encryption type should be used for interoperability with older levels of Kerberos Version 5.

ENCTYPE_DES_CBC_MD5 (x'00000003') MD5 checksum with DES encryption.

pre_auth_types (Input)
An array of preauthentication types to be used. The last entry in the array must be **KRB5_PADATA_NONE** (x'00000000'). If **NULL** is specified for this parameter, no preauthentication is done unless required by KDC policy. If multiple preauthentication types are specified, the KDC is supposed to accept the request as long as it recognizes at least one of the preauthentication types. Early implementations of the KDC did not follow this rule and will fail the request if the first preauthentication type is not recognized. The following preauthentication type may be specified:

KRB5_PADATA_ENC_TIMESTAMP (x'00000002') Encrypted timestamp preauthentication. This preauthentication type should be used for interoperability with a Kerberos KDC.

key (Input)
The key to be used. The default key table is used if **NULL** is specified for this parameter.

ccache (Input)
The credentials cache handle. The initial ticket ise stored in the credentials cache for later use by the application. The credentials are not stored if **NULL** is specified for this parameter.

creds (Input/Output)
The credentials that are used to obtain the initial ticket. The *client* and *server* fields must be set. The *endtime* field may be set to explicitly specify the ticket lifetime or it may be set to zero to use the default ticket lifetime. The *renew_till* field must be set if a renewable ticket is being requested. The *starttime* field must be set if a postdated ticket is being requested.

Upon completion of the request, **creds** are updated with the initial ticket, the session key, and the client address list. The **krb5_free_cred_contents()** or **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed.

ret_as_reply (Output)
The KDC reply. Specify **NULL** for this parameter if the KDC reply is not needed. The **krb5_free_kdc_rep()** routine should be called to release the reply when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_krbhst()—Get List of KDC Hosts

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_krbhst(
    krb5_context     context,
    krb5_const krb5_data *   realm,
    char ***       hostlist);
```

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

The **krb5_get_krbhst()** function returns a list of Kerberos Key Distribution Center (KDC) server hosts for a Kerberos realm. The list is obtained from the *[realms]* section of the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

realm (Input)
The Kerberos realm.

hostlist (Output)
The KDC host list. The last entry in the list is a **NULL** pointer. The **krb5_free_krbhst()** routine should be called to release the host list when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_get_server_rcache()—Generate Replay Cache for Server Use

Syntax

```
#include <krb5.h>

krb5_error_code krb5_get_server_rcache(
    krb5_context     context,
    krb5_const krb5_data *   piece);
    krb5_rcache *      ret_rcache);
```

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

The **krb5_get_server_rcache()** function generates a unique replay cache name and then opens the replay cache. The *piece* parameter is used to differentiate this replay cache from others currently in use on the system by the same user. The generated cache name is in the form **rc_piece_uid** and uses the default replay cache type.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the replay cache file	*X
Parent directory of the replay cache file, if **KRB5RCACHEDIR** is specified and if the replay cache file is being created	*WX
Replay cache file	*RW

Parameters

context (Input)
The Kerberos context.

piece (Input)
The unique portion of the replay cache name. It should consist of displayable characters.

ret_rcache (Output)
The replay cache handle. The **krb5_rc_close()** routine should be called to close the replay cache when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The replay cache is initialized if it cannot be recovered. The clock skew value is obtained from the Kerberos context if it is necessary to initialize the cache.

API introduced: V5R1

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

---

krb5_init_context()—Create and Initialize a Kerberos Context

Syntax

```
#include <krb5.h>

krb5_error_code krb5_init_context(
    krb5_context *     context);
```

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

The **krb5_init_context()** function creates a new Kerberos context and initializes it with default values obtained from the Kerberos configuration file. Each application needs at least one Kerberos context. A context may be shared by multiple threads within the same process. Use the **krb5_free_context()** routine to release the context when it is no longer needed.

Authorities

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

Parameters

context (Output)
The handle for the Kerberos context.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_add_entry()—Add New Entry to Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_add_entry(
    krb5_context       context,
    krb5_keytab      ktid,
    krb5_keytab_entry *    entry);
```

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

The **krb5_kt_add_entry()** function adds a new entry to a key table. No checking is done for duplicate entries. The key table type must support write operations.

Authorities

Object Referred to	Data Authority Required
All directories in the path name	*X
Keytab file	*RW

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

entry (Input)
The entry to be added to the key table.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. It is not necessary to add multiple entries to the key table for keys that use the same key generation algorithm. For example, encryption types **ENCTYPE_DES_CBC_CRC** and **ENCTYPE_DES_CBC_MD5** both generate a 56-bit DES key using the same algorithm. It is necessary to store only a single entry in the key table specifying one of these encryption types. The **krb5_kt_get_entry()** routine then returns this key table entry when either of these encryption types is specified.

API introduced: V5R1

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

---

krb5_kt_close()—Close Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_close(
    krb5_context       context,
    krb5_keytab      ktid);
```

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

The **krb5_kt_close()** function closes a key table. The key table handle may not be used once this routine completes.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_default()—Resolve Default Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_default(
    krb5_context       context,
    krb5_keytab *      ktid);
```

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

The **krb5_kt_default()** function resolves the default key table and returns a handle that can be used to access the table. This is equivalent to calling the **krb5_kt_resolve()** routine with the name returned by the **krb5_kt_default_name()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktid (Output)
The key table handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_default_name()—Get Default Key Table Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_default_name(
    krb5_context       context,
    char *       name,
    int        name_size);
```

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

The **krb5_kt_default_name()** function returns the name of the default key table for the current user. If the **KRB5_KTNAME** environment variable is set, this is the name of the default key table. Otherwise, the key table name is obtained from the *default_keytab_name* entry in the *[libdefaults]* section of the Kerberos configuration file. If this entry is not defined, the default key table name is **/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab**.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

name (Output)
The key table name.

name_size (Input)
The size of the buffer pointed to by the *name* parameter. The size must be large enough to contain the key table name and the trailing delimiter. One way to do this is to allocate the buffer to be **MAX_KEYTAB_NAME_LENGTH** (256) +1 bytes.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_end_seq_get()—End Sequential Reading of Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_end_seq_get(
    krb5_context       context,
    krb5_keytab      ktid,
    krb5_kt_cursor *     cursor);
```

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

The **krb5_kt_end_seq_get()** function ends the sequential reading of the key table and releases the cursor. The cursor may not be used once **krb5_kt_end_seq_get()** has completed.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

cursor (Input/Output)
The cursor created by the **krb5_kt_start_seq_get()** routine.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

Top | "Network Authentication Service APIs," on page 1
Security APIs | UNIX-Type APIs
APIs by category

krb5_kt_free_entry()—Free Storage Assigned to Key Table Entry

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_free_entry(
    krb5_context       context,
    krb5_keytab_entry *    entry);
```

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

The **krb5_kt_free_entry()** function releases the storage assigned to a key table entry. It does not free the krb5_keytab_entry structure itself.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

entry (Input)
The key table entry.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

krb5_kt_get_entry()—Get Entry from Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_get_entry(
    krb5_context       context,
    krb5_keytab      ktid,
    krb5_principal     principal,
    krb5_kvno        vno,
    krb5_enctype     enctype,
    krb5_keytab_entry *    entry);
```

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

The **krb5_kt_get_entry()** function returns an entry from the key table.

Authorities

Object Referred to	Data Authority Required
All directories in the path name	*X
Keytab file	*RW

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

principal (Input)
The principal.

vno (Input)
The key version number for the key to be retrieved. Specify a version number of zero to retrieve the key with the highest version number.

enctype (Input)
The key encryption type. Specify zero as the encryption type if the encryption type does not matter.

entry (Output)
The contents of the key table entry. The **krb5_kt_free_entry()** routine should be called to release the entry contents when they are no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The entry returned is the first one found in the key table that matches the requested principal and version, and uses a compatible encryption type. For example, an entry that uses **ENCTYPE_DES_CBC_MD5** is compatible with a requested encryption type of **ENCTYPE_DES_CBC_CRC**.

API introduced: V5R1

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

---

krb5_kt_get_name()—Get Key Table Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_get_name(
    krb5_context       context,
    krb5_keytab      ktid,
    char *       name,
    int        name_size);
```

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

The **krb5_kt_get_name()** function returns the name of the key table in the application-provided buffer supplied in the *name* parameter. The returned name includes the key table type prefix.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

name (Output)
The key table name.

name_size (Input)
The size of the buffer pointed to by the *name* parameter. The size must be large enough to contain the key table name and the trailing delimiter. This may be done by allocating the buffer to be **MAX_KEYTAB_NAME_LENGTH** (256) +1 bytes.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_get_type()—Get Key Table Type

Syntax

```
#include <krb5.h>

char * krb5_kt_get_type(
    krb5_context       context,
    krb5_keytab      ktid);
```

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

The **krb5_kt_get_type()** function returns the key table type.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

Return Value

This function returns the key table type. This is a read-only value and must not be freed by the application. For example, the character string ″FILE″ or ″WRFILE″ might be returned.

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

---

krb5_kt_next_entry()—Get Next Entry from Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_next_entry(
    krb5_context       context,
    krb5_keytab      ktid,
    krb5_keytab_entry *    entry,
    krb5_kt_cursor *     cursor);
```

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

The **krb5_kt_next_entry()** function reads the next entry from the key table and returns it to the application. The **krb5_kt_start_seq_get()** routine must be called to begin the sequential read operation. The **krb5_kt_next_entry()** routine then is called repeatedly to read table entries. Finally, the **krb5_kt_end_seq_get()** routine is called when no more entries are to be read.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

entry (Output)
The contents of the table entry. The **krb5_kt_free_entry()** routine should be called to release the entry contents when they are no longer needed.

cursor (Input/Output)
The cursor created by the **krb5_kt_start_seq_get()** routine. The cursor is updated upon successful completion of this routine.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_read_service_key()—Get Service Key from Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_read_service_key(
    krb5_context       context,
    krb5_pointer     keytab_name,
    krb5_principal     principal,
    krb5_kvno        vno,
    krb5_enctype     enctype,
    krb5_keyblock **     key);
```

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

The **krb5_kt_read_service_key()** function returns the service key from the key table.

Authorities

Object Referred to	Data Authority Required
All directories in path	*X
Keytab file	*R

Parameters

context (Input)
The Kerberos context.

keytab_name (Input)
The key table name. If a **NULL** address is specified, the default key table is used.

principal (Input)
The service principal.

vno (Input)
The key version number for the key to be retrieved. Specify a version number of zero to retrieve the key with the highest version number.

enctype (Input)
The key encryption type. Specify an encryption type of zero if the encryption type does not matter.

key (Output)
The retrieved key. The **krb5_free_keyblock()** routine should be called to release the key when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_register()—Register New Key Table Type

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_register(
    krb5_context       context,
    krb5_kt_ops *      ops);
```

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

The **krb5_kt_register()** function registers a new key table type. An error is returned if the key table type has already been registered. Once the new type is registered, it can be used by any thread in the current process and activation group. The type is not known outside the current process and activation group, and is no longer registered when the application ends.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ops (Input)
The key table operations vector. This vector defines the routines that are called to perform the various key table operations for the new type.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_remove_entry()—Remove Entry from Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_remove_entry(
    krb5_context       context,
    krb5_keytab      ktid,
    krb5_keytab_entry *    entry);
```

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

The **krb5_kt_remove_entry()** function removes an entry from a key table. The key table type must support write operations.

Authorities

Object Referred to	Data Authority Required
All directories in the path name	*X
Keytab file	*RW

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

entry (Input)
The entry to be removed from the key table.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_kt_resolve()—Resolve Key Table Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_resolve(
    krb5_context       context,
    krb5_const char *      keytab_name,
    krb5_keytab *      ktid);
```

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

The **krb5_kt_resolve()** function resolves a key table name and returns a handle that can be used to access the table.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

keytab_name (Input)
The key table name in the format ″type:name″. The type must be a registered key table type and the name must uniquely identify a particular key table of the specified type.

ktid (Output)
The key table handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos runtime supports two key table types: **FILE** and **WRFILE**. Additional key table types can be registered by the application by calling the **krb5_kt_register()** routine. If no type is specified, the default is **FILE**.

API introduced: V5R1

---

krb5_kt_start_seq_get()—Start Sequentially Retrieving Entries from Key Table

Syntax

```
#include <krb5.h>

krb5_error_code krb5_kt_start_seq_get(
    krb5_context       context,
    krb5_keytab      ktid,
    krb5_kt_cursor *     cursor);
```

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

The **krb5_kt_start_seq_get()** function starts sequentially retrieving entries from the key table. The **krb5_kt_next_entry()** routine is called repeatedly to retrieve each successive table entry. The **krb5_kt_end_seq_get()** routine is called at the completion of the read operation.

Authorities

Object Referred to	Data Authority Required
All directories in the path name	*X
Keytab file	*RW

Parameters

context (Input)
The Kerberos context.

ktid (Input)
The key table handle.

cursor (Output)
The cursor. The **krb5_kt_end_seq_get()** routine should be called to release the cursor at the completion of the sequential read operation.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The key table is locked when the **krb5_kt_start_seq_get()** routine is called and remains locked until the **krb5_kt_end_seq_get()** routine is called. Write access to the key table by other processes and threads is blocked until the table is unlocked. After the **krb5_kt_start_seq_get()** routine has been called, the current thread may not call any other key table functions except **krb5_kt_next_entry()** and **krb5_kt_end_seq_get()** for the specified table.

API introduced: V5R1

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

---

krb5_md5_crypto_compat_ctl()—Set Compatibility Mode for MD5 Checksum Generation

Syntax

```
#include <krb5.h>

void krb5_md5_crypto_compat_ctl(
    krb5_boolean   compat_mode);
```

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

The **krb5_md5_crypto_compat_ctl()** function sets the compatibility mode for the MD5 DES checksum generation.

Authorities

No authorities are required.

Parameters

compat_mode (Input)
The compatibility mode. It is specified as either **TRUE** or **FALSE**.

Return Value

This function does not return a value.

Error Messages

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

Usage Notes

1. Early beta levels of Kerberos Version 5 computed the MD5 DES checksum incorrectly. Enabling the compatibility mode causes the Kerberos runtime to generate the MD5 DES checksum the same way, while disabling the compatibility mode causes the Kerberos runtime to generate the checksum correctly.
2. This routine sets the MD5 compatibility mode for the entire process and overrides the compatibility mode set by the *rsa_md5_des_compat* entry in the Kerberos configuration file.

API introduced: V5R1

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

---

krb5_mk_error()—Create Kerberos KRB_ERROR Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_mk_error(
    krb5_context       context,
    krb5_const krb5_error *    dec_err,
    krb5_data *      enc_err);
```

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

The **krb5_mk_error()** function creates a Kerberos KRB_ERROR message. This message is then sent to the remote partner instead of sending a reply message. For example, if an error is detected while processing an AP_REQ message, the application returns a KRB_ERROR message instead of an AP_REP message.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

dec_err (Input)
The krb5_error structure to be encoded.

enc_err (Output)
The encoded krb5_error structure as a byte stream. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_mk_priv()—Create Kerberos KRB_PRIV Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_mk_priv(
    krb5_context       context,
    krb5_auth_context      auth_context,
    krb5_const krb5_data *   userdata,
    krb5_data *      out_data,
    krb5_replay_data *     replay_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes" on page 129.

The **krb5_mk_priv()** function creates a Kerberos KRB_PRIV message using data supplied by the application. The **krb5_mk_priv()** routine is similar to the **krb5_mk_safe()** routine, but the message is encrypted and integrity-protected rather than just integrity-protected. The **krb5_rd_priv()** routine decrypts and validates the message integrity.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

userdata (Input)
The application data for the KRB_PRIV message.

out_data (Output)
The KRB_PRIV message. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

replay_data (Output)
Replay information returned to the caller. This parameter is required if the **KRB5_AUTH_CONTEXT_RET_TIME** (x'00000002') or **KRB5_AUTH_CONTEXT_RET_SEQUENCE** (x'00000008') flag is set in the authentication context. Otherwise, **NULL** may be specified for this parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The authentication context specifies the checksum type, the data encryption type, the keyblock used to seed the checksum, the addresses of the sender and receiver, and the replay cache.
2. Use the **krb5_auth_con_setrcache()** routine to set the replay cache in the authentication context.
3. The local address in the authentication context is used to create the KRB_PRIV message and must be present. The remote address is optional. Use the **krb5_auth_con_genaddrs()** routine or a combination of the **krb5_auth_con_setaddrs()** and the **krb5_auth_con_setports()** routines to set the addresses in the authentication context. If the remote address is set, then the local address also must be set in the authentication context that is used for the **krb5_rd_priv()** routine. If port numbers are set, then they also must be set in the authentication context used for the **krb5_rd_priv()** routine.
4. The authentication context flags determine whether sequence numbers or timestamps should be used to identify the message. Use the **krb5_auth_con_set_flags()** routine to set the authentication context flags.
5. The encryption type is taken from the keyblock in the authentication context. If the initial vector has been set in the authentication context, it is used as the initialization vector for the encryption (if the encryption type supports initialization) and its contents are replaced with the last block of encrypted data upon return. Use the **krb5_auth_con_setivector()** routine or the **krb5_auth_con_initvector()** routine to modify the initial vector in the authentication context.
6. If timestamps are used (**KRB5_AUTH_CONTEXT_DO_TIME** (x'00000001') is set), an entry describing the message is entered in the replay cache so the caller can detect if this message is sent back by an attacker. An error is returned if the authentication context does not specify a replay cache.
7. If sequence numbers are used (**KRB5_AUTH_CONTEXT_DO_SEQUENCE** (x'00000004') or **KRB5_AUTH_CONTEXT_RET_SEQUENCE** (x'00000008') is set), the local sequence number in the authentication context is placed in the protected message as its sequence number.
8. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_mk_rep()—Create Kerberos AP_REP Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_mk_rep(
    krb5_context       context,
    krb5_auth_context      auth_context,
    krb5_data *      out_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes" on page 130.

The **krb5_mk_rep()** function creates a Kerberos AP_REP message using information in the authentication context. An AP_REP message is returned to the partner application after processing an AP_REQ message

received from the partner application. The information in the authentication context is set by the **krb5_rd_req()** routine when it processes the AP_REQ message.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context.

out_data (Output)
The AP_REP message. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_mk_req()—Create Kerberos AP_REQ Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_mk_req(
    krb5_context       context,
    krb5_auth_context *    auth_context,
    krb5_const krb5_flags    ap_req_options,
    char *       service,
    char *       hostname,
    krb5_data *      in_data,
    krb5_ccache      ccache,
    krb5_data *      out_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See “Usage Notes” on page 132.

The **krb5_mk_req()** function creates a Kerberos AP_REQ message. The checksum of the input data is included in the authenticator that is part of the AP_REQ message. This message is then sent to the partner application, which calls the **krb5_rd_req()** routine to extract the application data after validating the authenticity of the message. The checksum method set in the authentication context is used to generate the checksum.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context. A new authentication context is created and returned in this parameter if the value is **NULL**.

ap_req_options (Input)
The request options as follows:

AP_OPTS_USE_SESSION_KEY (x'40000000')	Use session key instead of server key. The credentials must include a ticket that is encrypted in the session key.
AP_OPTS_MUTUAL_REQUIRED (x'20000000')	Mutual authentication required.
AP_OPTS_USE_SUBKEY (x'00000001')	Generate a subsession key from the current session key obtained from the credentials.

service (Input)
The name of the service.

hostname (Input)
The host name that identifies the desired service instance.

in_data (Input)
The application data's checksum that is to be included in the authenticator. Specify **NULL** for this parameter if no checksum is to be included in the authenticator.

ccache (Input)
The credentials cache that is to be used to obtain credentials to the desired service.

out_data (Output)
The AP_REQ message. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The **krb5_sname_to_principal()** routine is called to convert the *service* and *hostname* parameters to a Kerberos principal. The **krb5_get_host_realm()** routine is called to convert the *hostname* parameter to a Kerberos realm. If the credentials cache does not already contain a service ticket for the target server, the Kerberos protocol runtime issues a default TGS request to obtain the credentials and store them in the cache.
2. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_mk_req_extended()—Create Kerberos AP_REQ Message Using Supplied Credentials

Syntax

```
#include <krb5.h>

krb5_error_code krb5_mk_req_extended(
    krb5_context       context,
    krb5_auth_context *    auth_context,
    krb5_const krb5_flags    ap_req_options,
    krb5_data *      in_data,
    krb5_creds *     in_creds,
    krb5_data *      out_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See “Usage Notes” on page 133.

The **krb5_mk_req_extended()** function creates a Kerberos AP_REQ message using supplied credentials. It is similar to the **krb5_mk_req()** routine, but the caller passes the actual credentials as a parameter rather than letting the Kerberos runtime construct the credentials. The checksum of the input data is included in the authenticator that is part of the AP_REQ message. This message is then sent to the partner application, which calls the **krb5_rd_req()** routine to extract the application data after validating the authenticity of the message. The checksum method set in the authentication context is used to generate the checksum.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context. A new authentication context is created and returned in this parameter if the value is **NULL**.

ap_req_options (Input)
The request options as follows:

AP_OPTS_USE_SESSION_KEY (x'40000000')	Use session key instead of server key. The credentials must include a ticket that is encrypted in the session key.
AP_OPTS_MUTUAL_REQUIRED (x'20000000')	Mutual authentication required.
AP_OPTS_USE_SUBKEY (x'00000001')	Generate a subsession key from the current session key obtained from the credentials.

in_data (Input)
The application data's checksum that is to be included in the authenticator.

in_creds (Input)
The credentials for the specified service.

out_data (Output)
The AP_REQ message. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

Return Value
If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes
1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_mk_safe()—Create Kerberos KRB_SAFE Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_mk_safe(
    krb5_context       context,
```

```
krb5_auth_context      auth_context,
krb5_const krb5_data *   userdata,
krb5_data *      out_data,
krb5_replay_data *     replay_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_mk_safe()** function creates a Kerberos KRB_SAFE message using data supplied by the application. Messages created by the **krb5_mk_safe()** routine are integrity-protected. This routine returns an error if the message has been modified.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context.

userdata (Input)
The application data for the KRB_SAFE message.

out_data (Output)
The KRB_SAFE message. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

replay_data (Output)
Replay information returned to the caller. This parameter is required if the **KRB5_AUTH_CONTEXT_RET_TIME** (x'00000002') or **KRB5_AUTH_CONTEXT_RET_SEQUENCE** (x'00000008') flag is set in the authentication context. Otherwise, **NULL** may be specified for this parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The authentication context specifies the checksum type, the keyblock used to seed the checksum, the addresses of the sender and receiver, and the replay cache.
2. Use the **krb5_auth_con_setrcache()** routine to set the replay cache in the authentication context.
3. The local address in the authentication context is used to create the KRB_SAFE message and must be present. The remote address is optional. Use the **krb5_auth_con_genaddrs()** routine or a combination of the **krb5_auth_con_setaddrs()** and the **krb5_auth_con_setports()** routines to set the addresses in the authentication context. If the remote address is set, then the local address also must be set in the authentication context that is used for the **krb5_rd_safe()** routine. If port numbers are set, then they also must be set in the authentication context used for the **krb5_rd_safe()** routine.

4. The authentication context flags determine whether sequence numbers or timestamps should be used to identify the message. Use the **krb5_auth_con_set_flags()** routine to set the authentication context flags.
5. If timestamps are used (**KRB5_AUTH_CONTEXT_DO_TIME** (x'00000001') is set), an entry describing the message is entered in the replay cache so the caller can detect if this message is sent back by an attacker. An error is returned if the authentication context does not specify a replay cache.
6. If sequence numbers are used (**KRB5_AUTH_CONTEXT_DO_SEQUENCE** (x'00000004') or **KRB5_AUTH_CONTEXT_RET_SEQUENCE** (x'00000008') is set), the local sequence number in the authentication context is placed in the protected message as its sequence number.
7. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_os_hostaddr()—Get Network Addresses Used by Specific Host System

Syntax

```
#include <krb5.h>

krb5_error_code krb5_os_hostaddr(
    krb5_context       context,
    krb5_const char *      host,
    krb5_address ***     addrs);
```

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

The **krb5_os_hostaddr()** function returns the network addresses used by a specific host system. At the present time, only the **AF_INET** address family is supported, and the **gethostbyname_r()** system function is used to search for the addresses assigned to the specified host.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

host (Input)
The name of the host system. The name must be acceptable for use with the **gethostbyname_r()** system function.

addrs (Output)
An array of krb5_address pointers. The last entry in the array is a **NULL** pointer. The **krb5_free_addresses()** routine should be called to release the address array when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_os_localaddr()—Return Network Addresses Used by Local System

Syntax

```
#include <krb5.h>

krb5_error_code krb5_os_localaddr(
    krb5_context       context,
    krb5_address ***     addrs);
```

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

The **krb5_os_localaddr()** function returns the network addresses used by the local system. At the present time, only the **AF_INET** address family is supported.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

addrs (Output)
An array of krb5_address pointers. The last entry in the array is a **NULL** pointer. The **krb5_free_addresses()** routine should be called to release the address array when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_parse_name()—Create Kerberos Principal from Text String

Syntax

```
#include <krb5.h>

krb5_error_code krb5_parse_name(
    krb5_context       context,
    krb5_const char *      name,
    krb5_principal *     principal);
```

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

The **krb5_parse_name()** routine converts a text string into a Kerberos principal. The string must be in the format **name@realm**. If the realm is not specified, the default realm is used. Each forward slash in the name starts a new name component unless it is escaped by preceding the forward slash with a backward slash. Forward slashes in the realm are not treated as component separators and are copied unchanged.

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.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

name (Input)
The string to be parsed. The string must be in the format **name@realm**.

principal (Output)
The Kerberos principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_principal_compare()—Compare Two Kerberos Principals

Syntax

```
#include <krb5.h>

krb5_boolean krb5_principal_compare(
    krb5_context   context,
    krb5_const_principal princ1,
    krb5_const_principal princ2);
```

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

The **krb5_principal_compare()** function allows an application to compare two Kerberos principals.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

princ1 (Input)
The first principal to be compared.

princ2 (Input)
The second principal to be compared.

Return Value

TRUE	The principal names are the same.
FALSE	The principal names are different.

Error Messages

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

API introduced: V5R1

krb5_random_confounder()—Create Random Confounder

Syntax

```
#include <krb5.h>

krb5_error_code krb5_random_confounder(
    krb5_context       context,
    int        buffer_size,
    krb5_pointer     output_buffer);
```

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

The **krb5_random_confounder()** function creates a random value that can be used as a confounder when encrypting data. A confounder is used to initialize the encryption block chaining value so the encrypted result is different each time a data value is encrypted, even when the data value and encryption key are not changed.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

buffer_size (Input)
The size of the output buffer.

output_buffer (Output)
The buffer to receive the confounder.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_close()—Close Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_close(
    krb5_context     context,
    krb5_rcache      rcache);
```

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

The **krb5_rc_close()** function closes a replay cache. The cache handle may not be used once this routine completes.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_default()—Resolve Default Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_default(
    krb5_context     context,
    krb5_rcache *      rcache);
```

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

The **krb5_rc_default()** function resolves the default replay cache and returns a handle that can be used to access the table. This is equivalent to calling the **krb5_rc_resolve()** routine with the name returned by the **krb5_rc_default_name()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Output)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_default_name()—Get Default Replay Cache Name

Syntax

```
#include <krb5.h>

char * krb5_rc_default_name(
    krb5_context     context);
```

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

The **krb5_rc_default_name()** function returns the name of the default replay cache for the current user. The **KRB5RCACHENAME** environment variable defines the default replay cache name.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

Return Value

The name of the default replay cache for the current user or **NULL** if the default name has not been set. The return value is the address of a read-only string and must not be freed by the application.

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

---

krb5_rc_destroy()—Delete Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_destroy(
    krb5_context     context,
    krb5_rcache      rcache);
```

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

The **krb5_rc_destroy()** function closes and deletes a replay cache. The cache handle may not be used once this routine completes.

Authorities

When the replay cache is of type "dfl" (see **krb5_rc_resolve()** for more information on replay cache types), the default behavior is that the replay cache file is created in the /QIBM/UserData/OS400/NetworkAuthentication/replay directory. The placement of the replay cache file can be changed by setting the KRB5RCACHEDIR or KRB5RCACHENAME environment variable, or by specifying a different path with the **krb5_rc_resolve()** function.

If the default directory is not used, the following authorities are required:

Object Referred to	Data Authority Required	Object Authority Required
Each directory in the path name preceding the replay cache file	*X	None
Parent directory of the replay cache file	*WX	None
Replay cache file	*RW	*OBJEXIST

If the default directory is used, the following authorities are required:

Object Referred to	Data Authority Required
Each directory in the path name preceding the replay cache file	*X
Replay cache file	*RW

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_rc_expunge()—Delete Expired Entries from Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_expunge(
    krb5_context     context,
    krb5_rcache      rcache);
```

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

The **krb5_rc_expunge()** function deletes expired entries from the replay cache. The entry lifespan is set by the **krb5_rc_initialize()** routine. This routine should be called periodically to purge the replay cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

krb5_rc_free_entry_contents()—Free Storage Associated with Replay Cache Entry

Syntax

```
#include <krb5.h>

void krb5_rc_free_entry_contents(
    krb5_context     context,
    krb5_donot_replay *    entry);
```

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

The **krb5_rc_free_entry_contents()** function releases the storage associated with a replay cache entry. The krb5_donot_replay structure itself will not be released.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

entry (Input)
The entry to be released.

Return Value

This routine does not return a value.

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

---

krb5_rc_get_lifespan()—Get Authenticator Lifespan for Entries in Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_get_lifespan(
    krb5_context     context,
    krb5_rcache      rcache,
    krb5_deltat *      span);
```

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

The **krb5_rc_get_lifespan()** function returns the authenticator lifespan for entries in the replay cache. This lifespan was set by the **krb5_rc_initialize()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

span (Output)
The authenticator lifespan in seconds.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_get_name()—Get Replay Cache Name

Syntax

```
#include <krb5.h>

char * krb5_rc_get_name(
    krb5_context     context,
    krb5_rcache      rcache);
```

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

The **krb5_rc_get_name()** function returns the replay cache name. The returned name does not include the replay cache type prefix.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

Return Value

The **krb5_rc_get_name()** routine returns the name of the replay cache. This is a read-only value and must not be freed by the application.

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

---

krb5_rc_get_type()—Get Replay Cache Type

Syntax

```
#include <krb5.h>

char * krb5_rc_get_type(
    krb5_context     context,
    krb5_rcache      rcache);
```

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

The **krb5_rc_get_type()** function returns the replay cache type. For example, the character strings ″dfl″ or ″mem″ might be returned.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

Return Value

The **krb5_rc_get_type()** routine returns returns the replay cache type. This is a read-only value and must not be freed by the application.

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

---

krb5_rc_initialize()—Initialize Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_initialize(
    krb5_context     context,
    krb5_rcache      rcache,
    krb5_deltat      span);
```

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

The **krb5_rc_initialize()** function initializes a replay cache. Any existing cache entries are deleted. The authenticator lifespan indicates how long an authenticator remains valid. Once an authenticator has expired, its replay cache entry can be deleted by calling the **krb5_rc_expunge()** routine.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

span (Input)
The authenticator lifespan in seconds.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_recover()—Recover Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_recover(
    krb5_context     context,
    krb5_rcache      rcache);
```

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

The **krb5_rc_recover()** function recovers a replay cache after the application has been restarted. Either **krb5_rc_recover()** or **krb5_rc_initialize()** must be called before any replay entries can be added to the replay cache.

Authorities

Object Referred to	Data Authority Required
Each directory in the path name preceding the replay cache file	*X
Replay cache file	*RW

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_register_type()—Define New Replay Cache Type

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_register_type(
    krb5_context     context,
    krb5_rc_ops *      ops);
```

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

The **krb5_rc_register_type()** function allows an application to define a new replay cache type. An error is returned if the replay cache type has already been registered. Once the new type is registered, it can be used by any thread in the current process and activation group. The type is not known outside the current process and activation group, and is no longer registered when the application ends.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ops (Input)
The replay cache operations vector. This vector defines the routines that will be called to perform the various replay cache operations for the new type.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_rc_resolve()—Resolve Replay Cache Name

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_resolve(
    krb5_context     context,
    krb5_rcache *      rcache,
    char *       name);
```

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

The **krb5_rc_resolve()** resolves a replay cache name and returns a handle that can be used to access the cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Output)
The replay cache handle.

name (Input)
The replay cache name in the format "type:name". The type must be a registered replay cache type and the name must uniquely identify a particular replay cache of the specified type.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos runtime supports two replay cache types: **dfl** and **mem**. Additional replay cache types can be registered by the application by calling the **krb5_rc_register_type()** routine. If no type is specified, the default is **dfl**.
2. After successfully calling **krb5_rc_resolve()**, the application should call either the **krb5_rc_recover()** or the **krb5_rc_initialize()** routine.

API introduced: V5R1

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

---

krb5_rc_store()—Store New Entry in Replay Cache

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rc_store(
    krb5_context     context,
    krb5_rcache      rcache,
    krb5_donot_replay *    replay);
```

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

The **krb5_rc_store()** function stores a new entry in the replay cache after verifying that the entry is not already in the cache.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

rcache (Input)
The replay cache handle.

replay (Input)
The replay entry.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The **krb5_auth_to_rep()** routine can be used to create a replay entry from a Kerberos authenticator. The **krb5_rc_expunge()** routine should be called periodically to purge expired entries from the replay cache.

API introduced: V5R1

---

krb5_rd_error()—Process Kerberos KRB_ERROR Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rd_error(
    krb5_context       context,
    krb5_const krb5_data *   enc_err,
    krb5_error **      dec_err);
```

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

The **krb5_rd_error()** function processes a Kerberos KRB_ERROR message created by the **krb5_mk_error()** routine and returns a krb5_error structure.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

enc_err (Input)
The error message created by the **krb5_mk_error()** routine.

dec_err (Output)
The decoded error message. The **krb5_free_error()** routine should be called to release the krb5_error structure when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

---

krb5_rd_priv()—Process Kerberos KRB_PRIV Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rd_priv(
    krb5_context       context,
    krb5_auth_context      auth_context,
    krb5_const krb5_data *   in_data,
    krb5_data *      out_data,
    krb5_replay_data *     replay_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See “Usage Notes” on page 153.

The **krb5_rd_priv()** function processes a Kerberos KRB_PRIV message and extracts the application data after verifying its integrity. If timestamps are being used, the message is stored in the replay cache associated with the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context.

in_data (Input)
The buffer containing the KRB_PRIV message.

out_data (Output)
The application data. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

replay_data (Output)
Replay information returned to the caller. This parameter is required if the **KRB5_AUTH_CONTEXT_RET_TIME** (x’00000002’) or **KRB5_AUTH_CONTEXT_RET_SEQUENCE** (x’00000008’) flag is set in the authentication context. Otherwise, **NULL** may be specified for this parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The keyblock used for decrypting data and verifying message integrity is obtained from the authentication context. The first non-NULL keyblock is used by checking the local_subkey, remote_subkey, or keyblock, in that order. If the initialization vector in the authentication context has been set, it is used to initialize the decryption (if the encryption type supports initialization) and its contents are replaced with the last block of encrypted data in the message upon return. Use the **krb5_auth_con_setivector()** routine or the **krb5_auth_con_initvector()** routine to modify the initial vector in the authentication context.
2. The remote address in the authentication context must be present. It specifies the address of the sender. Use the **krb5_auth_con_genaddrs()** routine or the **krb5_auth_con_setaddrs()** routine to set the remote address. If the port number was set in the authentication context used for the **krb5_mk_priv()** routine, then the port number also must be set in the authentication context used for the **krb5_rd_priv()** routine. An error is returned if the address or port in the message does not match the remote address or port in the authentication context.
3. The local address in the authentication context is optional. If it is present, then it must match the receiver address in the message. Otherwise, the receiver address in the message must match one of the local addresses returned by the **krb5_os_localaddr()** routine. If the port number was set in the authentication context used for the **krb5_mk_priv()** routine, then both the local address and the local port must be set in the authentication context used for the **krb5_rd_priv()** routine. Use the **krb5_auth_con_genaddrs()** routine or a combination of the **krb5_auth_con_setaddrs()** and **krb5_auth_con_setports()** routines to set the local address and local port in the authentication context.
4. Use the **krb5_auth_con_setrcache()** routine to set the replay cache in the authentication context.
5. If timestamps are being used (**KRB5_AUTH_CONTEXT_DO_TIME** (x'00000001') is set in the authentication context), the timestamp in the message must be within the Kerberos clock skew for the current time. In addition, the message must not be found in the replay cache obtained from the authentication context. Use the **krb5_auth_con_setflags()** routine to set the **KRB5_AUTH_CONTEXT_DO_TIME** flag.
6. If message sequence numbers are being used (**KRB5_AUTH_CONTEXT_DO_SEQUENCE** is set in the authentication context), the remote sequence number in the authentication context must match the sequence number in the message. Use the **krb5_auth_con_setflags()** routine to set the **KRB5_AUTH_CONTEXT_DO_SEQUENCE** flag.
7. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

krb5_rd_rep()—Process Kerberos AP_REP Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rd_rep(
    krb5_context       context,
    krb5_auth_context      auth_context,
    krb5_const krb5_data *   in_data,
    krb5_ap_rep_enc_part **    reply);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See "Usage Notes."

The **krb5_rd_rep()** function processes a Kerberos AP_REP message created by the **krb5_mk_rep()** routine. The authentication context is updated with sequencing information obtained from the reply message.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context.

in_data (Input)
The buffer containing the AP_REP message.

reply (Output)
The decrypted reply data. The **krb5_free_ap_rep_enc_part()** routine should be called to release the reply when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

krb5_rd_req()—Process Kerberos AP_REQ Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rd_req(
    krb5_context       context,
    krb5_auth_context *    auth_context,
    krb5_const krb5_data *   in_data,
    krb5_const_principal   server,
    krb5_keytab      keytab,
    krb5_flags *     ap_req_options,
    krb5_ticket **     ticket);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See “Usage Notes” on page 156.

The **krb5_rd_req()** function processes a Kerberos AP_REQ message generated by the partner application. The authenticator is extracted, validated, and stored in the authentication context. If the *server* parameter is not **NULL** and no replay cache is associated with the authentication context, the Kerberos protocol runtime creates a replay cache and stores the cache handle in the authentication context.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context. A new authentication context is created and returned in this parameter if the value is **NULL**.

in_data (Input)
The buffer containing the AP_REQ message.

server (Input)
The server name. The server principal in the AP_REQ must be the same as the principal specified by this parameter. Specify **NULL** if any server principal is acceptable.

keytab (Input)
The key table that contains the server key. The default key table is used if **NULL** is specified for this parameter.

ap_req_options (Output)
The options from the AP_REQ message. Specify **NULL** for this parameter if the options are not needed.

ticket (Output)
The ticket from the AP_REQ message. Specify **NULL** for this parameter if the ticket is not needed. The **krb5_free_ticket()** routine should be called to release the ticket when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Authorities

No authorities are required.

Error Messages

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

Usage Notes

1. If the authentication context contains a keyblock, it is used to decrypt the ticket in the AP_REQ message. This is useful for user-to-user authentication. If the authentication context does not contain a keyblock, the key table specified on the function call is used to obtain the decryption key.
2. The client in the authenticator must match the client in the ticket. If the remote address has been set in the authentication context, the request must have come from that address. If a replay cache handle is stored in the authentication context, the new authenticator is stored in the cache after checking for replay.
3. If no errors are detected, the authenticator, subsession key, and remote sequence number are stored in the authentication context. If **AP_OPTS_MUTUAL_REQUIRED** (x'20000000') is specified in the AP_REQ message, the local sequence number is XORed with the remote sequence number.
4. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

---

krb5_rd_req_verify()—Process and Verify Kerberos AP_REQ Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rd_req_verify(
    krb5_context     context,
    krb5_auth_context *               auth_context,
    const krb5_data *                 in_data,
    const krb5_data *                 appl_data,
    krb5_const_principal              server,
    krb5_keytab                       keytab,
    krb5_flags *                      ap_req_options,
    krb5_ticket **                    ticket);
```

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

The **krb5_rd_req_verify()** function processes an AP_REQ message generated by the partner application and verifies the application data checksum contained in the authenticator. The authenticator is extracted,

validated, and stored in the authentication context. If the server parameter is not NULL and no replay cache is associated with the authentication context, the Kerberos runtime will create a replay cache and store the cache handle in the authentication context.

Authorities

None.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context. A new authentication context will be created and returned if this parameter is NULL.

in_data (Input)
The buffer containing the AP_REQ message.

appl_data (Input)
The application data to be verified. The checksum is computed for the supplied data and compared to the checksum obtained from the authenticator. Specify NULL if the checksum is not to be verified.

server (Input)
The server name. The server principal in the AP_REQ must be the same as the principal specified by this parameter. Specify NULL if any server principal is acceptable.

keytab (Input)
The key table which contains the server key. The default key table will be used if NULL is specified for this parameter.

ap_req_options (Output)
The options returned from the AP_REQ message. Specify NULL for this parameter if the options are not needed.

ticket (Output)
The ticket returned from the AP_REQ message. Specify NULL for this parameter if the ticket is not needed. The **krb5_free_ticket()** routine should be called to release the ticket when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. If the authentication context contains a keyblock, it will be used to decrypt the ticket in the AP_REQ message. This is useful for user-to-user authentication. If the authentication context does not contain a keyblock, the key table specified on the function call will be used to obtain the decryption key.

2. The client in the authenticator must match the client in the ticket. If the remote address has been set in the authentication context, the request must have come from that address. If a replay cache handle is stored in the authentication context, the new authenticator is stored in the cache after checking for replay.
3. If no errors are detected, the authenticator, subsession key, and remote sequence number are stored in the authentication context. If AP_OPTS_MUTUAL_REQUIRED is specified in the AP_REQ message, the local sequence number is XORed with the remote sequence number

API introduced: V5R2

---

krb5_rd_safe()—Process Kerberos KRB_SAFE Message

Syntax

```
#include <krb5.h>

krb5_error_code krb5_rd_safe(
    krb5_context       context,
    krb5_auth_context      auth_context,
    krb5_const krb5_data *   in_data,
    krb5_data *      out_data,
    krb5_replay_data *     replay_data);
```

Service Program Name: QSYS/QKRBGSS
Default Public Authority: *USE
Threadsafe: Conditional. See “Usage Notes” on page 159.

The **krb5_rd_safe()** function processes a Kerberos KRB_SAFE message and extracts the application data after verifying its integrity. If timestamps are being used, the message is stored in the replay cache associated with the authentication context.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

auth_context (Input)
The authentication context.

in_data (Input)
The buffer containing the KRB_SAFE message.

out_data (Output)
The application data. The **krb5_free_data_contents()** routine should be called to release the storage pointed to by the *data* field of the krb5_data structure when it is no longer needed.

replay_data (Output)
Replay information returned to the caller. This parameter is required if the **KRB5_AUTH_CONTEXT_RET_TIME** (x’00000002’) or **KRB5_AUTH_CONTEXT_RET_SEQUENCE** (x’00000008’) flag is set in the authentication context. Otherwise, **NULL** may be specified for this parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. The keyblock that is used for verifying message integrity is obtained from the authentication context. The first non-NULL keyblock is used by checking the local_subkey, remote_subkey, or keyblock, in that order.
2. The remote address in the authentication context must be present. It specifies the address of the sender. Use the **krb5_auth_con_genaddrs()** routine or the **krb5_auth_con_setaddrs()** routine to set the remote address. If the port number was set in the authentication context used for the **krb5_mk_safe()** routine, then the port number also must be set in the authentication context used for the **krb5_rd_safe()** routine. An error is returned if the address in the message does not match the remote address in the authentication context.
3. The local address in the authentication context is optional. If it is present, then it must match the receiver address in the message. Otherwise, the receiver message in the message must match one of the local addresses returned by the **krb5_os_localaddr()** routine. If the port number was set in the authentication context used for the **krb5_mk_safe()** routine, then both the local address and the local port must be set in the authentication context used for the **krb5_rd_priv()** routine. Use the **krb5_auth_con_genaddrs()** routine or a combination of the **krb5_auth_con_setaddrs()** and **krb5_auth_con_setports()** routines to set the local address and local port in the authentication context.
4. Use the **krb5_auth_con_setrcache()** routine to set the replay cache in the authentication context.
5. If message sequence numbers are being used (**KRB5_AUTH_CONTEXT_DO_SEQUENCE** (x'00000004') is set in the authentication context), the remote sequence number in the authentication context must match the sequence number in the message. Use the **krb5_auth_con_setflags()** routine to set the **KRB5_AUTH_CONTEXT_DO_SEQUENCE** flag.
6. If timestamps are being used (**KRB5_AUTH_CONTEXT_DO_TIME** (x'00000001') is set in the authentication context), the timestamp in the message must be within the Kerberos clock skew for the current time. In addition, the message must not be found in the replay cache obtained from the authentication context. Use the **krb5_auth_con_setflags()** routine to set the **KRB5_AUTH_CONTEXT_DO_TIME** flag.
7. The Kerberos protocol runtime provides no concurrency control for the authentication context. If the application wants to use the same authentication context in multiple threads, it is the responsibility of the application to serialize access to the authentication context so that only a single thread is accessing the authentication context at any time. Because message sequence numbers are contained in the authentication context, this serialization needs to be extended to encompass the message exchange between the two applications. Otherwise, message sequence errors are liable to occur if the messages are delivered out of sequence.

API introduced: V5R1

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

krb5_realm_compare()—Compare Realm Names of Two Principals

Syntax

```
#include <krb5.h>

krb5_boolean krb5_realm_compare(
    krb5_context   context,
    krb5_const_principal princ1,
    krb5_const_principal princ2);
```

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

The **krb5_const_principal()** function compares the realm names of two principals.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

princ1 (Input)
The first principal to be compared.

princ2 (Input)
The second principal to be compared.

Return Value

TRUE	The realm names are equal.
FALSE	The realm names are different.

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

---

krb5_recvauth()—Process an Authentication Message Stream

Syntax

```
#include <krb5.h>

krb5_error_code krb5_recvauth(
    krb5_context     context,
    krb5_auth_context *                auth_context,
    krb5_pointer                       socket,
    char *                             appl_version,
    krb5_principal                     server,
    krb5_int32                         flags,
    krb5_keytab                        keytab,
    krb5_ticket **                     ticket);
```

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

The **krb5_recvauth()** function processes an authentication message stream generated by the **krb5_sendauth()** routine. It receives the authentication message and sends the authentication response using the socket descriptor supplied by the application. The application is responsible for establishing the connection before calling the **krb5_recvauth()** routine.

The **krb5_recvauth()** routine processes an AP_REQ message generated by the partner application. The authenticator is extracted, validated, and stored in the authentication context. If the server parameter is not NULL and no replay cache is associated with the authentication context, the Kerberos runtime will create a replay cache and store the cache handle in the authentication context..

Authorities

None.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context. A new authentication context will be created and returned in this parameter if the value is NULL.

socket (Input)
The address of a socket descriptor. This descriptor must represent a TCP stream connection and not a UDP datagram connection.

appl_version (Input)
The application version message. An error will be returned if this application version message does not match the application version message s upplied by the sender. Specify NULL for this parameter if the application version message does not need to be verified. The supplied application version message will be converted to the network code page before comparing it with the sender's application version message.

server (Input)
The server name. The server principal in the AP_REQ must be the same as the principal specified by this parameter. Specify NULL if any server principal is acceptable.

flags (Input)
Specifies flags for the **krb5_recvauth()** routine. There are currently no defined flags.

keytab (Input)
The key table which contains the server key. The default key table will be used if NULL is specified for this parameter.

ticket (Output)
The ticket returned from the AP_REQ message. Specify NULL for this parameter if the ticket is not needed. The **krb5_free_ticket()** routine should be called to release the ticket when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. If the authentication context contains a keyblock, it will be used to decrypt the ticket in the AP_REQ message. This is useful for user-to-user authentication. If the authentication context does not contain a keyblock, the key table specified on the function call will be used to obtain the decryption key.
2. The client in the authenticator must match the client in the ticket. If the remote address is set in the authentication context, the address list in the ticket must either include that address or must be a null list. If a replay cache handle is stored in the authentication context, the new authenticator is stored in the cache after checking for replay.
3. If no errors are detected, the authenticator, subsession key, and remote sequence number are stored in the authentication context. If AP_OPTS_MUTUAL_REQUIRED is specified in the AP_REQ message, the local sequence number is XORed with the remote sequence number.

API introduced: V5R2

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

---

krb5_sendauth()—Send an Authentication Message Stream

Syntax

```
#include <krb5.h>

krb5_error_code krb5_sendauth(
    krb5_context     context,
    krb5_auth_context *               auth_context,
    krb5_pointer                      socket,
    char *                            appl_version,
    krb5_principal                    client,
    krb5_principal                    server,
    krb5_int32                        app_req_options,
    krb5_data *                       appl_data,
    krb5_creds *                      in_creds,
    krb5_ccache                       ccache,
    krb5_error **                     error,
    krb5_ap_rep_enc_part **           rep_result,
    krb5_creds **                     out_creds)
```

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

The **krb5_sendauth()** function generates an authentication message stream for processing by the **krb5_recvauth()** routine. It sends the authentication message and receives the authentication response using the socket descriptor supplied by the application. The application is responsible for establishing the connection before calling the **krb5_sendauth()** routine. The **krb5_sendauth()** routine generates an AP_REQ message. The checksum of the application data is included in the authenticator which is part of the AP_REQ message. This message is then sent to the partner application, which calls the **krb5_recvauth()** routine to validate the authenticity of the message. The checksum method set in the authentication context is used to generate the checksum.

Authorities

None.

Parameters

context (Input)
The Kerberos context.

auth_context (Input/Output)
The authentication context. A new authentication context will be created and returned in this parameter if the value is NULL.

socket (Input)
The address of a socket descriptor. This descriptor must represent a TCP stream connection and not a UDP datagram connection.

appl_version (Input)
The application version message. An error will be returned if this application version message does not match the application version message supplied by the receiver. The supplied application version message will be converted to the network code page before being sent to the partner application.

client (Input)
The client name. This parameter is ignored if a non-NULL value is supplied for the 'in_creds' parameter. The client name is obtained from the credentials cache if this parameter is NULL.

server (Input)
The server name. This parameter is ignored if a non-NULL value is provided for the 'in_creds' parameter.

ap_req_options (Input)
Request options as follows:

AP_OPTS_USE_SESSION_KEY	Use session key instead of server key for the service ticket. The credentials must include a ticket which is encrypted in the session key.
AP_OPTS_MUTUAL_REQUIRED	Mutual authentication required.
AP_OPTS_USE_SUBKEY	Generate a subsession key from the current session key obtained from the credentials.

appl_data (Input)
The application data whose checksum is to be included in the authenticator. Specify NULL for this parameter if no checksum is to be included in the authenticator.

in_creds (Input)
The credentials for the specified service. The 'client' and 'server' parameters are ignored if a non-NULL value is provided for the 'in_creds' parameter. In this case, the client and server names must be set in the input credentials. The service ticket may be supplied as part of the input credentials by setting a non-zero ticket length value. If the service ticket is not supplied as part of the input credentials, the Kerberos runtime will obtain a service ticket using the ticket-granting ticket retrieved from the credentials cache.

When the Kerberos runtime obtains the service ticket, additional fields are checked in the input credentials. The second_ticket field must be set if the service ticket is to be encrypted in a session key. The ticket expiration time can be set to override the default expiration time. The key encryption type can be set to override the default ticket encryption type.

ccache (Input)
The credentials cache which is to be used to obtain credentials to the desired service. The

credentials cache is not used when the service ticket is supplied as part of the input credentials. The default credentials cache will be used if this parameter is NULL.

error (Output)
The KRB_ERROR message returned if an authentication error is reported by the partner application. The **krb5_free_error()** routine should be called to release the error message when it is no longer needed. Specify NULL for this parameter if the error message is not needed.

rep_result (Output)
The decrypted reply data returned from the AP_REP message. The **krb5_free_ap_rep_enc_part()** routine should be called to release the reply data when it is no longer needed. Specify NULL for this parameter if the reply data is not needed. A reply is available only if AP_OPTS_MUTUAL_REQUIRED is specified in the request options.

out_creds (Output)
The service ticket returned. The **krb5_free_creds()** routine should be called to release the credentials when they are no longer needed. Specify NULL for this parameter if the service ticket is not needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R2

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

---

krb5_set_config_files()—Set Files to be Processed for Kerberos Configuration Requests

Syntax

```
#include <krb5.h>

krb5_error_code krb5_set_config_files(
    krb5_context       context,
    krb5_const char **     names);
```

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

The **krb5_set_config_files()** function specifies the names of the files to be processed to obtain the Kerberos configuration. These files replace the configuration files that were used to create the Kerberos context. Changing the configuration files does not affect context values that have already been set from the old configuration files.

Authorities

Object Referred to	Authority Required
Each directory in the path names preceding each of the configuration files specified	*X
Configuration files	*R

Parameters

context (Input)
The Kerberos context.

names (Input)
An array of file names. The last entry in the array must be a **NULL** pointer.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_set_default_in_tkt_ktypes()—Set Default Encryption Types to Request Initial Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_set_default_in_tkt_ktypes(
    krb5_context     context,
    krb5_const krb5_enctype *    ktypes);
```

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

The **krb5_set_default_in_tkt_ktypes()** function sets the default encryption types to be used when requesting an initial ticket from the Kerberos server. The first encryption type specified is used for generating random keys, so it must be an encryption type that is supported by the Kerberos server. The encryption types specified override any values specified by the *default_tkt_enctypes* entry in the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktypes (Input)
An array of krb5_enctype values to be used when requesting an initial ticket. The last element in the array must be set to **ENCTYPE_NULL** (x'00000000'). The following symbolic definitions are provided for specifying the encryption types:

ENCTYPE_DES_CBC_CRC (x'00000001')	DES encryption with a CRC checksum
ENCTYPE_DES_CBC_MD5 (x'00000003')	DES encryption with an MD5 checksum
ENCTYPE_DES_CBC_RAW (x'00000004')	DES encryption with no checksum

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. To interoperate with older Kerberos Version 5 servers, you should specify **ENCTYPE_DES_CBC_CRC** as the first encryption type.

API introduced: V5R1

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

---

krb5_set_default_realm()—Set Default Realm for Local System

Syntax

```
#include <krb5.h>

krb5_error_code krb5_set_default_realm(
    krb5_context     context,
    char *       realm);
```

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

The **krb5_set_default_realm()** function sets the default realm for the specified Kerberos context. This overrides the default realm set by the Kerberos configuration file. The realm set by **krb5_set_default_realm()** applies only to the Kerberos context specified by the *context* parameter.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

realm (Input)
The name for the default realm.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_set_default_tgs_ktypes()—Set Default Encryption Types to Request Service Ticket

Syntax

```
#include <krb5.h>

krb5_error_code krb5_set_default_tgs_ktypes(
    krb5_context     context,
    krb5_const krb5_enctype *    ktypes);
```

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

The **krb5_set_default_tgs_ktypes()** function sets the default encryption types to be used when requesting a service ticket from the Kerberos server. The first encryption type specified is used for generating random keys, so it must be an encryption type that is supported by the Kerberos server. The encryption types specified override any values specified by the *default_tgs_enctypes* entry in the Kerberos configuration file.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

ktypes (Input)
An array of krb5_enctype values to be used when requesting a service ticket. The last element in the array must be set to **ENCTYPE_NULL** (x'00000000'). The following symbolic definitions are provided for specifying the encryption types:

ENCTYPE_DES_CBC_CRC (x'00000001')	DES encryption with a CRC checksum
ENCTYPE_DES_CBC_MD5 (x'00000003')	DES encryption with an MD5 checksum
ENCTYPE_DES_CBC_RAW (x'00000004')	DES encryption with no checksum

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

Usage Notes

1. To interoperate with older Kerberos Version 5 servers, you should specify **ENCTYPE_DES_CBC_CRC** as the first encryption type.

API introduced: V5R1

---

krb5_sname_to_principal()—Convert Service Name to a Kerberos Principal

Syntax

```
#include <krb5.h>

krb5_error_code krb5_sname_to_principal(
    krb5_context     context,
    krb5_const char *      hostname,
    krb5_const char *      sname,
    krb5_int32       type,
    krb5_principal *     ret_princ);
```

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

The **krb5_sname_to_principal()** function converts a service name and a host name to a Kerberos principal. The principal name is in the format **sname/hostname@realm**. The realm name that corresponds to the host name is obtained by calling the **krb5_get_host_realm()** routine.

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.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

hostname (Input)
The host containing the desired service instance. The local host is used if **NULL** is specified for this parameter.

sname (Input)
The service name. The service name is set to the character string ″host″ if **NULL** is specified for this parameter.

type (Input)
The type of host name provided as follows:

KRB5_NT_SRV_HST (x'00000003')	A DNS host name has been provided. The Kerberos runtime looks up the address assigned to the host name and then does a reverse search to get the primary host name for that address. The resulting host name then is converted to lowercase.
KRB5_NT_UNKNOWN (x'00000000')	The host name type is unknown. No translation is performed on the specified host name and it is used as is.

ret_princ (Output)
The generated principal. The **krb5_free_principal()** routine should be called to release the principal when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_svc_get_msg()—Get Printable Text Message Corresponding to Kerberos Error Code

Syntax

```
#include <krb5.h>

krb5_error_code krb5_svc_get_msg(
    krb5_ui_4        error_code,
    char **        msg_text);
```

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

The **krb5_svc_get_msg()** function returns a printable text message corresponding to a Kerberos error code. This allows the application to log the error or display it to the user.

Authorities

No authorities are required.

Parameters

error_code (Input)
The Kerberos error code.

msg_text (Output)
The character string describing the error code. The **krb5_free_string()** routine should be called to release the character string when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_timeofday()—Get Current Time of Day in Seconds since the Epoch

Syntax

```
#include <krb5.h>

krb5_error_code krb5_timeofday(
    krb5_context     context,
    krb5_timestamp *     seconds);
```

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

The **krb5_timeofday()** function returns the current time of day in seconds since the epoch (January 1, 1970). The returned time is calculated using the **gettimeofday()** routine. This means that the returned time is Coordinated Universal Time. The returned time also is adjusted for changes made to the software clock by the **adjtime()** or **settimeofday()** routines.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

seconds (Output)
The number of seconds since the epoch.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_unparse_name()—Convert a Kerberos Principal to Text String

Syntax

```
#include <krb5.h>

krb5_error_code krb5_unparse_name(
    krb5_context       context,
    krb5_const_principal   principal,
    char **        name);
```

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

The **krb5_unparse_name()** function creates a text string from a Kerberos principal. The string is in the format **name@realm**, with the name components separated by forward slashes. If a forward slash occurs within a name component, it is escaped in the generated string by preceding the forward slash with a backward slash.

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.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

principal (Input)
The principal to be converted.

name (Output)
The text string for the principal in the format **name@realm**. The **krb5_free_string()** routine should be called to release the returned string when it is no longer needed.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

---

krb5_unparse_name_ext()—Convert a Kerberos Principal Extended to Text String

Syntax

```
#include <krb5.h>

krb5_error_code krb5_unparse_name_ext(
    krb5_context       context,
    krb5_const_principal   principal,
    char **        name,
    int *        size);
```

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

The **krb5_unparse_name_ext()** function creates a text string from a Kerberos principal. It is similar to the **krb5_unparse_name()** function, but it allows the application to avoid the overhead of repeatedly allocating the output string when a large number of conversions need to be performed. The string is in the format **name@realm**, with the name components separated by forward slashes. If a forward slash occurs within a name component, it is escaped in the generated string by preceding the forward slash with a backward slash.

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.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

principal (Input)
The principal to be converted.

name (Input/Output)
The text string for the principal in the format **name@realm**. The **krb5_free_string()** routine should be called to release the returned string when it is no longer needed. If the name parameter contains a **NULL** address upon entry, **krb5_unparse_name_ext()** allocates a new buffer and returns the address in the *name* parameter and the size in the *size* parameter. Otherwise, the *name* parameter must contain the address of an existing buffer and the *size* parameter must contain the size of this buffer. The **krb5_unparse_name_ext()** reallocates the buffer if necessary and returns the updated values in the *name* and *size* parameters.

size (Input/Output)
The size of the buffer specified by the *name* parameter.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

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

---

krb5_us_timeofday()—Get Current Time of Day in Seconds and Microseconds since the Epoch

Syntax

```
#include <krb5.h>

krb5_error_code krb5_us_timeofday(
    krb5_context     context,
    krb5_timestamp *     seconds);
    krb5_int32 *     useconds);
```

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

The **krb5_us_timeofday()** function returns the current time of day in seconds and microseconds since the epoch (January 1, 1970). The returned time is calculated using the **gettimeofday()** routine. This means that the returned time is Coordinated Universal Time. The returned time also is adjusted for changes made to the software clock by the **adjtime()** or **settimeofday()** routines.

Authorities

No authorities are required.

Parameters

context (Input)
The Kerberos context.

seconds (Output)
The seconds' portion of the result.

useconds (Output)
The microseconds' portion of the result.

Return Value

If no errors occur, the return value is 0. Otherwise, a Kerberos error code is returned.

Error Messages

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

API introduced: V5R1

---

qkrb_add_kt_entry()—Add Keytab Entry

Syntax

```
#include <krb5.h>

void qkrb_add_kt_entry(char *     keytab,
                       char *     principal,
                       char *     password,
                       krb5_kvno  version);
```

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

The **qkrb_add_kt_entry()** function allows you to add a keytab entry to a keytab file for a specified principal name. If a principal name and version number match an existing keytab entry, the entry is replaced.

Authorities and Locks

Object	Authority Required
All directories in the path to the keytab file	*X
Keytab file	*RW

Parameters

keytab (Input)
The name of the keytab file to receive the new entry.

NULL The new entry will be placed in the default keytab file.

principal (Input)
The principal name for the new keytab entry.

Note: If the realm name is not included in the specified principal name, the default realm will be appended to the name.

password (Input)
The password value for the new keytab entry.

version (Input)
The version number for the new keytab entry.

0 Create the new entry with the default version value.

Note: Keytab entries are identifed by their principal name's. The first time a keytab entry is

created, the default version value is 1. If keytab entries exists for the specified principal, the default is to add 1 to the largest version number of the existing entries.

Error Messages

Message ID	Error Message Text
CPE4ABB E	Network Authentication Service failed with return code &1.

Example

See Code disclaimer information for information pertaining to code examples.

The following example will add a keytab entry to the default keytab file.

```
#include <krb5.h>
#include <string.h.h>

int main(int argc, char *argv[])
{
    /* Add a keytab entry to the default keytab file for the          */
    /* specified principal.                                           */
    /* This program accepts 2 parameters:                             */
    /*     1: Pointer to the principal name for the new keytab entry  */
    /*     2: Pointer to the password for the new keytab entry        */

    char *principal;
    char *password;

    /* Copy the address of the principal and the password to local    */
    /* variables.                                                     */
    principal = argv[1];
    password  = argv[2];

    /* Create the keytab entry for the specified principal.           */
    /*   NOTE:  When the first parameter, keytab, is set to NULL the  */
    /*          default keytab file is used.  The default file is     */
    /*          commonly:                                             */
    /* /QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab  */
    qkrb_add_kt_entry(NULL, principal, password);

    return;
}
```

API introduced: V5R3

Top | Security APIs | APIs by category

qkrb_count_kt_entries()—Count Keytab Entries

Syntax

```
#include <krb5.h>

void qkrb_count_kt_entries(char *      keytab,
                           char *      principal,
                           krb5_kvno   version,
                           int  *      count);
```

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

The **qkrb_count_kt_entries()** function allows you to obtain the total count of entries in a keytab file or count the number of keytab entries there are for a particular principal.

Authorities and Locks

Object	Authority Required
All directories in the path to the keytab file	*X
Keytab file	*R

Parameters

keytab (Input)
The name of the keytab file that will be searched.

NULL The default keytab file will be searched.

principal (Input)
The principal name of the keytab entries being counted.

NULL Include keytab entries with any principal name in the count.

Note: If the realm name is not included in the specified principal name, the default realm will be appended to the name.

version (Input)
The version number of the keytab entries being counted.

0 Include keytab entries with any version in the count.

count (Output)
The total number of entries in the keytab file that meet the search criteria specified in the principal name and version number parameters.

Error Messages

Message ID	Error Message Text
CPE3C3C E	Value for parameter & not valid.
CPE4ABB E	Network Authentication Service failed with return code &1.

Example

See Code disclaimer information for information pertaining to code examples.

The following example will count the number of keytab entries in the default keytab file.

```
#include <krb5.h>
#include <string.h.h>
#include <stdio.h.h>

int main(int argc, char *argv[])
{
    /* Count the number of keytab entries in the default keytab       */
    /* file for the specified principal name and version number.      */
    /* This program accepts 2 parameters:                             */
    /*     1: Pointer to keytab entry's principal name                */
    /*     2: Keytab entry's version number                           */

    char       *principal;
```

```
    krb5_kvno  version;
    int        return_count;

    /* Copy the address of the principal name and the value of the    */
    /* version number to local variables.                             */
    principal = argv[1];
    version   = atoi(argv[2]);

    /* Clear the count.                                               */
    return_count = 0;

    /* Count the keytab entries.                                      */
    /*   NOTE:  When the first parameter, keytab, is set to NULL the  */
    /*          default keytab file is used.  The default keytab file */
    /*          is commonly:                                          */
    /* /QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab  */
    qkrb_count_kt_entries(NULL, principal, version, &return_count);

    /* Print the count.                                               */
    printf ("%s Principal: \n", principal);
    printf ("%i Version number: \n", (int)version);
    printf ("%i Number of keytab entries: \n", return_count);

    return;
}
```

API introduced: V5R3

Top | Security APIs | APIs by category

---

qkrb_remove_kt_entry()—Remove Keytab Entry

Syntax

```
#include <krb5.h>

void qkrb_remove_kt_entry(char *      keytab,
                          char *      principal,
                          krb5_kvno   version);
```

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

The **qkrb_remove_kt_entry()** function allows you to remove keytab entries from a keytab file for a specified principal.

Authorities and Locks

Object	Authority Required
All directories in the path to the keytab file	*X
Keytab file	*RW

Parameters

keytab (Input)
The name of the keytab file from which the entry is removed.

NULL The keytab entries will be removed from the default keytab file.

principal (Input)
The principal name of the keytab entry being removed.

Note: If the realm name is not included in the specified principal name, the default realm will be appended to the name.

version (Input)
The version number of the keytab entry being removed.

0 Remove all versions of keytab entries from the keytab file.

Error Messages

Message ID	Error Message Text
CPE4ABB E	Network Authentication Service failed with return code &1.

Example

See Code disclaimer information for information pertaining to code examples.

The following example will remove a keytab entry from the default keytab file.

```
#include <krb5.h>
#include <string.h.h>

int main(int argc, char *argv[])
{
    /* Remove all the keytab entries from the default keytab file     */
    /* for the specified principal.                                   */
    /* This program accepts 1 parameter:                              */
    /*   1: Pointer to the principal name of the entry being removed. */

    char *principal;

    /* Copy the address of the principal to a local variable.         */
    principal = argv[1];

    /* Remove all versions of the principal's keytab entries from the */
    /* default keytab file.                                           */
    /*   NOTES: When the first parameter, keytab, is set to NULL the  */
    /*          default keytab file is used.  The default file is     */
    /*          commonly:                                             */
    /* /QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab  */
    /*                                                                */
    /*          When the third parameter, version, is set to 0 all    */
    /*          versions of the keytab entries will be removed.       */
    qkrb_remove_kt_entry(NULL, principal, 0);

    return;
}
```

API introduced: V5R3

Top | Security 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 information 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.

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:

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

Programming Interface Information

This Application Programming Interfaces (API) 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 Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

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, and 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