Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 111.

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.

© Copyright International Business Machines Corporation 1998, 2006. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Secure sockets APIs .................................. 1
APIs .................................................. 2
i5/OS Global Secure Toolkit (GSKit) APIs ........ 2
gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment ........................................... 3
  Parameters ........................................... 3
  Authorities ......................................... 4
  Return Value ....................................... 4
  Error Conditions ................................. 5
  Usage Notes ....................................... 5
  Related Information ............................... 6

gsk_attribute_get_cert_info()—Get information about a local or partner certificate ......................................................... 6
  Parameters ........................................... 7
  Authorities ......................................... 7
  Return Value ....................................... 7
  Error Conditions ................................. 8
  Usage Notes ....................................... 8
  Related Information ............................... 9

gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment> .................................. 10
  Parameters ........................................... 10
  Authorities ......................................... 12
  Return Value ....................................... 12
  Error Conditions ................................. 12
  Usage Notes ....................................... 13
  Related Information ............................... 13

gsk_attribute_get_numeric_value()—Get numeric information about a secure session or an SSL environment ............................. 14
  Parameters ........................................... 14
  Authorities ......................................... 14
  Return Value ....................................... 14
  Error Conditions ................................. 15
  Usage Notes ....................................... 15
  Related Information ............................... 16

gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment ........................................... 16
  Parameters ........................................... 17
  Authorities ......................................... 17
  Return Value ....................................... 17
  Error Conditions ................................. 18
  Usage Notes ....................................... 18
  Related Information ............................... 19

gsk_attribute_setCallback()—Set callback pointers to routines in the user application ......................................................... 20
  Parameters ........................................... 20
  Authorities ......................................... 22
  Return Value ....................................... 22
  Error Conditions ................................. 22
  Usage Notes ....................................... 23
  Related Information ............................... 23

gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment ........................................... 23
  Parameters ........................................... 24
  Authorities ......................................... 26
  Return Value ....................................... 26
  Error Conditions ................................. 26
  Usage Notes ....................................... 27
  Related Information ............................... 27

gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment ........................................... 28
  Parameters ........................................... 28
  Authorities ......................................... 28
  Return Value ....................................... 28
  Error Conditions ................................. 29
  Usage Notes ....................................... 29
  Related Information ............................... 30

gsk_environment_close()—Close an SSL environment ......................................................... 30
  Parameters ........................................... 30
  Authorities ......................................... 30
  Return Value ....................................... 31
  Error Conditions ................................. 31
  Usage Notes ....................................... 31
  Related Information ............................... 31

gsk_environment_init()—Initialize an SSL environment ......................................................... 32
  Parameters ........................................... 32
  Authorities ......................................... 32
  Return Value ....................................... 32
  Error Conditions ................................. 33
  Usage Notes ....................................... 34
  Related Information ............................... 34

gsk_environment_open()—Get a handle for an SSL environment ......................................................... 34
  Parameters ........................................... 34
  Authorities ......................................... 35
  Return Value ....................................... 35
  Error Messages ..................................... 35
  Usage Notes ....................................... 35
  Related Information ............................... 36

gsk_secure_soc_close()—Close a secure session ......................................................... 36
  Parameters ........................................... 36
  Authorities ......................................... 36
  Return Value ....................................... 36
  Error Conditions ................................. 37
  Error Messages ..................................... 37
  Usage Notes ....................................... 37
  Related Information ............................... 37

gsk_secure_soc_init()—Negotiate a secure session ......................................................... 37
  Parameters ........................................... 38
  Authorities ......................................... 38
  Return Value ....................................... 38
  Error Conditions ................................. 39
  Usage Notes ....................................... 39
  Error Messages ..................................... 40

© Copyright IBM Corp. 1998, 2006
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Messages</td>
<td>87</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>87</td>
</tr>
<tr>
<td>Related Information</td>
<td>87</td>
</tr>
<tr>
<td>SSL_Perror()—Print SSL Error Message</td>
<td>87</td>
</tr>
<tr>
<td>Parameters</td>
<td>88</td>
</tr>
<tr>
<td>Authorities</td>
<td>88</td>
</tr>
<tr>
<td>Return Value</td>
<td>88</td>
</tr>
<tr>
<td>Error Conditions.</td>
<td>88</td>
</tr>
<tr>
<td>Error Messages</td>
<td>88</td>
</tr>
<tr>
<td>Related Information</td>
<td>88</td>
</tr>
<tr>
<td>Example</td>
<td>89</td>
</tr>
<tr>
<td>SSL_Read()—Receive Data from an SSL-Enabled Socket</td>
<td>89</td>
</tr>
<tr>
<td>Socket Descriptor</td>
<td>90</td>
</tr>
<tr>
<td>Parameters</td>
<td>90</td>
</tr>
<tr>
<td>Authorities</td>
<td>90</td>
</tr>
<tr>
<td>Return Value</td>
<td>90</td>
</tr>
<tr>
<td>Error Conditions.</td>
<td>91</td>
</tr>
<tr>
<td>Error Messages</td>
<td>92</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>92</td>
</tr>
<tr>
<td>Related Information</td>
<td>93</td>
</tr>
<tr>
<td>SSL_Strerror()—Retrieve SSL Runtime Error Message</td>
<td>93</td>
</tr>
<tr>
<td>Parameters</td>
<td>93</td>
</tr>
<tr>
<td>Authorities</td>
<td>94</td>
</tr>
<tr>
<td>Return Value</td>
<td>94</td>
</tr>
<tr>
<td>Error Conditions.</td>
<td>94</td>
</tr>
<tr>
<td>Error Messages</td>
<td>94</td>
</tr>
<tr>
<td>Related Information</td>
<td>94</td>
</tr>
<tr>
<td>Example</td>
<td>94</td>
</tr>
<tr>
<td>SSL_Write()—Write Data to an SSL-Enabled Socket</td>
<td>95</td>
</tr>
<tr>
<td>Descriptor</td>
<td>95</td>
</tr>
<tr>
<td>Parameters</td>
<td>96</td>
</tr>
<tr>
<td>Authorities</td>
<td>96</td>
</tr>
<tr>
<td>Return Value</td>
<td>96</td>
</tr>
<tr>
<td>Error Conditions.</td>
<td>97</td>
</tr>
<tr>
<td>Error Messages</td>
<td>98</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>98</td>
</tr>
<tr>
<td>Related Information</td>
<td>98</td>
</tr>
<tr>
<td>Concepts</td>
<td>98</td>
</tr>
<tr>
<td>Header Files for UNIX-Type Functions</td>
<td>99</td>
</tr>
<tr>
<td>Errno Values for UNIX-Type Functions</td>
<td>101</td>
</tr>
</tbody>
</table>

**Appendix. Notices** 111

- Programming Interface Information 112
- Trademarks 113
- Terms and Conditions 114
Secure sockets APIs

Secure sockets consists of the following APIs:

- "i5/OS Global Secure Toolkit (GSKit) APIs” on page 2
- "i5/OS Secure Sockets Layer (SSL_) APIs” on page 64
- Open SSL APIs

The i5/OS® Global Secure Toolkit (GSKit) and i5/OS SSL_ application programming interfaces (APIs) are a set of functions that, when used with the i5/OS sockets APIs, are designed to enable and facilitate secure communications between processes on a network. The GSK Secure Toolkit (GSKit) APIs are the preferred set of APIs to be used to securely enable an application using Secure Sockets Layer/Transport Layer Security (SSL/TLS). The SSL_ APIs also can be used to enable an application to use the SSL/TLS Protocol.

SSL provides communications privacy over an open communications network (that is, the Internet). The protocol allows client/server applications to communicate to prevent eavesdropping, tampering, and message forgery. The SSL protocol connection security has three basic properties:

- The connection is private. Encryption using secret keys is used to encrypt and decrypt the data. The secret keys are generated on a per SSL session basis using an SSL handshake protocol. An SSL handshake is a series of protocol packets sent in a particular sequence, which use asymmetric cryptography to establish an SSL session. Symmetric cryptography is used for application data encryption and decryption.
- The peer’s identity can be authenticated using asymmetric, or public key cryptography.
- The connection is reliable. Message transport includes a message integrity check using a keyed Message Authentication Code (MAC). Secure hash functions are used for MAC computations.

When creating ILE programs or service programs that use the i5/OS GSKit or SSL_ APIs, you do not need to explicitly bind to the secure sockets service program QSYS/QSOSSLSR because it is part of the system binding directory.

The GSKit and SSL_ API documentation describes the GSKit and SSL_ APIs only. This documentation does not include any information about how to configure or obtain any of the cryptographic objects, such as a key ring file or certificate, that are required to fully enable an application for SSL. Some cryptographic objects, such as certificate store files, are required parameters for GSKit and SSL_ APIs. Information on how to configure the cryptographic objects required for the i5/OS secure socket APIs, or how to configure a secure web server, which also uses the secure socket APIs, can be found using the following references:

- HTTP Server: Documentation

- Secure Sockets Layer (SSL) under the Security topic. Plan for enabling SSL discusses what you must install and configure before using secure sockets.
- Cryptographic Hardware topic.

For background information on GSKit and SSL_ APIs, see:

- Secure Sockets in the Sockets programming topic.
APIs

These are the APIs for this category.

i5/OS Global Secure Toolkit (GSKit) APIs

i5/OS™ GSKit APIs, when used in addition to the existing i5/OS Sockets APIs, provide the functions required for applications to establish secure communications. An application using GSKit for secure communications basically is a client/server application written using sockets.

The Global Secure Toolkit (GSKit) APIs are:

- **“gsk_attribute_get_buffer()”—Get character information about a secure session or an SSL environment** on page 3 (Get character information about a secure session or an SSL environment) is used to obtain specific character string information about a secure session or an SSL environment.
- **“gsk_attribute_get_cert_info()”—Get information about a local or partner certificate“ on page 6** (Get information about a local or partner certificate) is used to obtain specific information about either the server or client certificate for a secure session or an SSL environment.
- **“gsk_attribute_get_enum()”—Get enumerated information about a secure session or an SSL environment” on page 10** (Get enumerated information for a secure session or an SSL environment) is used to obtain values for specific enumerated data for a secure session or an SSL environment.
- **“gsk_attribute_get_numeric_value()”—Get numeric information about a secure session or an SSL environment” on page 14** (Get numeric information about a secure session or an SSL environment) is used to obtain specific numeric information about a secure session or an SSL environment.
- **“gsk_attribute_set_buffer()”—Set character information for a secure session or an SSL environment” on page 16** (Set character information for a secure session or an SSL environment) is used to set a specified buffer attribute to a value inside the specified secure session or SSL environment.
- **“gsk_attribute_set_callback()”—Set callback pointers to routines in the user application’ on page 20** (Set callback pointers to routines in the user application) is used to set callback callback pointers to routines in the user application.
- **“gsk_attribute_set_enum()”—Set enumerated information for a secure session or an SSL environment” on page 23** (Set enumerated information for a secure session or an SSL environment) is used to set a specified enumerated type attribute to an enumerated value in the secure session or SSL environment.
- **“gsk_attribute_set_numeric_value()”—Set numeric information for a secure session or an SSL environment” on page 28** (Set numeric information for a secure session or an SSL environment) is used to set specific numeric information for a secure session or an SSL environment.
- **“gsk_environment_close()”—Close an SSL environment” on page 30** (Close an SSL environment) is used to close the SSL environment and release all storage associated with the environment.
- **“gsk_environment_init()”—Initialize an SSL environment” on page 32** (Initialize an SSL environment) is used to initialize the SSL environment after any required attributes are set.
- **“gsk_environment_open()”—Get a handle for an SSL environment” on page 34** (Get a handle for an SSL environment) is used to get storage for the SSL environment.
- **“gsk_secure_soc_close()”—Close a secure session” on page 36** (Close a secure session) is used to close a secure session and free all the associated resources for that secure session.
- **“gsk_secure_soc_init()”—Negotiate a secure session” on page 37** (Negotiate a secure session) is used to negotiate a secure session, using the attributes set for the SSL environment and the secure session.
- **“gsk_secure_soc_misc()”—Perform miscellaneous functions for a secure session” on page 40** (Perform miscellaneous functions for a secure session) is used to perform miscellaneous functions for a secure session.
- **“gsk_secure_soc_open()”—Get a handle for a secure session” on page 43** (Get a handle for a secure session) is used to get storage for a secure session, set default values for attributes, and return a handle that must be saved and used on secure session-related function calls.
gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment

Syntax

```c
#include <gskssl.h>

int gsk_attribute_get_buffer(gsk_handle my_gsk_handle,
                             GSK_BUF_ID bufId,
                             const char **buffer,
                             int *bufSize);
```

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

The `gsk_attribute_get_buffer()` function is used to obtain specific character string information about a secure session or an SSL environment. It can be used to obtain values such as certificate store file, certificate store password, application ID, and ciphers.

Parameters

**my_gsk_handle (Input)**
- Indicates one of the following handles:
  - The handle for the secure session (`my_session_handle`)
  - The handle for the SSL environment (`my_env_handle`)
bufID (Input)
The following values can be used to retrieve information about the secure session or the SSL environment that is either defaulted or explicitly set:

- **GSK_KEYRING_FILE (201)** - buffer points to the name of the certificate store file being used for the SSL environment.
- **GSK_KEYRING_PW (202)** - buffer points to the password for the certificate store file being used for the SSL environment.
- **GSK_KEYRING_LABEL (203)** - buffer points to the certificate label associated with the certificate in the certificate store identified by **GSK_KEYRING_FILE** to be used for the secure session or SSL environment.
- **GSK_OS400_APPLICATION_ID (6999)** - buffer points to the application identifier being used for the SSL environment.
- **GSK_V2_CIPHER_SPECS (205)** - buffer points to the list of available SSL Version 2 ciphers to be used for the secure session or the SSL environment. See the usage notes in “**gsk_attribute_set_buffer()**—Set character information for a secure session or an SSL environment” on page 16 API for the format of the ciphers.
- **GSK_V3_CIPHER_SPECS (206)** - buffer points to the list of available SSL Version 3 or TLS Version 1 ciphers to be used for the secure session or the SSL environment. See the usage notes in “**gsk_attribute_set_buffer()**—Set character information for a secure session or an SSL environment” on page 16 API for the format of the ciphers.
- **GSK_CONNECT_SEC_TYPE (208)** - buffer points to a string containing "SSLV2," "SSLV3," or "TLSV1," depending on what was actually negotiated for use by the secure session.
- **GSK_CONNECT_CIPHER_SPEC (207)** - buffer points to a one- or two-character string describing the cipher specification negotiated for use by the secure session. See the usage notes in “**gsk_attribute_set_buffer()**—Set character information for a secure session or an SSL environment” on page 16 API for the format of the ciphers.
- **GSK_SID_VALUE (212)** - buffer points to a string containing the session ID (SID) used for the secure session.

buffer (Output)
The address of the location to place the pointer that will point to the buffer containing the requested information. The storage for this information was allocated by the system from user heap storage and will be freed by the **gsk_secure_soc_close()** API or the **gsk_environment_close()** API.

The data in the buffer is assumed to be represented in the CCSID (coded character set identifier) currently in effect for the job. If the CCSID of the job is 65535, this buffer is assumed to be represented in the default CCSID of the job.

bufSize (Output)
The address of the location to store the length of the requested information pointed to by buffer.

Authorities
No authorization is required.

Return Value
**gsk_attribute_get_buffer()**

returns an integer. Possible values are:

- **[GSK_OK]**
  
  gsk_attribute_get_buffer() was successful.
[GSK_ATTRIBUTE_INVALID_ID]
The specified bufID was not valid.

[GSK_INVALID_HANDLE]
The specified handle was not valid.

[GSK_AS400_ERROR_INVALID_POINTER]
The buffer or bufSize pointer is not valid.

[GSK_ERROR_UNSUPPORTED]
The bufID currently is not supported.

[GSK_ERROR_IO]
An error occurred in SSL processing. Check the errno value.

**Error Conditions**

When the `gsk_attribute_get_buffer()` API fails with return code [GSK_ERROR_IO], `errno` can be set to:

- [EINTR] Interrupted function call.
- [EDEADLK] Resource deadlock avoided.
- [ETERM] Operation terminated.

If an `errno` is returned that is not in this list, look in ["Errno Values for UNIX-Type Functions" on page 101] for a description of the `errno`.

**Usage Notes**

1. The following `GSK_BUF_ID` values may be retrieved from the SSL environment after `gsk_environment_open()`.

   - GSK_KEYRING_FILE
   - GSK_KEYRING_PW
   - GSK_KEYRING_LABEL
   - GSK_V2_CIPHER_SPECS
   - GSK_V3_CIPHER_SPECS

2. The following `GSK_BUF_ID` values may be retrieved from the secure session after `gsk_secure_soc_open()`.

   - GSK_KEYRING_LABEL
   - GSK_V2_CIPHER_SPECS
   - GSK_V3_CIPHER_SPECS
   - GSK_CONNECT_SEC_TYPE
   - GSK_CONNECT_CIPHER_SPEC

3. The following `GSK_BUF_ID` values are defaulted after `gsk_secure_soc_open()` and will be set for the secure session after `gsk_secure_soc_init()`.

   - GSK_CONNECT_SEC_TYPE
   - GSK_CONNECT_CIPHER_SPEC
   - GSK_SID_VALUE

4. The following `GSK_BUF_ID` values may be changed for the secure session after `gsk_secure_soc_misc()`, `gsk_secure_soc_read()` or `gsk_secure_soc_startRecv()` if an SSL Handshake happened under the context of those calls for the secure session.
5. You can reference the buffer pointer as long as the handle for the secure session or the SSL environment is still open.

6. The following GSK_BUF_ID values currently are not supported in the i5/OS implementation:

- GSK_KEYRING_STASH_FILE
- GSK_LDAP_SERVER
- GSK_LDAP_USER
- GSK_LDAP_USER_PW
- GSK_USER_DATA
- GSK_PKCS11_DRIVER_PATH
- GSK_PKCS11_TOKEN_LABEL
- GSK_PKCS11_TOKEN_PWD
- GSK_CSP_NAME

Related Information

- “gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment” on page 16
- “gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment” on page 10
- “gsk_attribute_get_numeric_value()—Get numeric information about a secure session or an SSL environment” on page 14
- “gsk_attribute_get_cert_info()—Get information about a local or partner certificate”
- “gsk_environment_close()—Close an SSL environment” on page 30
- “gsk_environment_init()—Initialize an SSL environment” on page 32
- “gsk_environment_open()—Get a handle for an SSL environment” on page 34
- “gsk_secure_soc_close()—Close a secure session” on page 36
- “gsk_secure_soc_init()—Negotiate a secure session” on page 37
- “gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40
- “gsk_secure_soc_open()—Get a handle for a secure session” on page 43
- “gsk_strerror()—Retrieve GSKit runtime error message” on page 62

API introduced: V5R1

```
gsk_attribute_get_cert_info()—Get information about a local or partner certificate

Syntax
#include <gskssl.h>

int gsk_attribute_get_cert_info(gsk_handle my_gsk_handle,
    GSK_CERT_ID certID,
    const gsk_cert_data_elem **certDataElem,
    int *certDataElemCount);
```
Service Program Name: QSYS/QSOSSLR
Default Public Authority: *USE
Threadsafe: Yes

The `gsk_attribute_get_cert_info()` function is used to obtain specific information about either the server or client certificate for a secure session or an SSL environment.

**Parameters**

**my_gsk_handle (Input)**
Indicates one of the following handles:

- The handle for the secure session. (my_session_handle)
- The handle for the SSL environment. (my_env_handle)

**certID (Input)**
Indicates one of the following:

- **GSK_LOCAL_CERT_INFO (701)** - Retrieve certificate data information for the local certificate that may be sent to the remote connection. This can be retrieved using the SSL environment handle or the secure session handle.
- **GSK_PARTNER_CERT_INFO (700)** - Retrieve certificate data information for the partner certificate that may have been received during the SSL handshake. This can only be retrieved using the secure session handle.

**certDataElem (Output)**
The address of a pointer to the certificate information returned from this function call. On output, `certDataElem` will contain the pointer to the information. The storage for this information was allocated by the system from user heap storage and will be freed by the `gsk_secure_soc_close()` API or the `gsk_environment_close()` API.

**certDataElemCount (Output)**
A pointer to an integer that will contain the number of certificate data elements returned from this function call.

**Authorities**
No authorization is required.

**Return Value**
`gsk_attribute_get_cert_info()` returns an integer. Possible values are:

**[GSK_OK]**
`gsk_attribute_get_cert_info()` was successful.

**[GSK_ATTRIBUTE_INVALID_ID]**
The specified `certID` was not valid.

**[GSK_INVALID_HANDLE]**
The handle passed in to this function was not valid.

**[GSK_INVALID_STATE]**
One of the following occurred:

- A SSL environment handle was specified with a `certID` of GSK_LOCAL_CERT_INFO before a `gsk_environment_init()` has been issued.
- A secure session handle was specified before a `gsk_secure_soc_init()` has been issued.

**[GSK_AS400_ERROR_INVALID_POINTER]**
The `certDataElem` or `certDataElemCount` pointer is not valid.
[GSK_INSUFFICIENT_STORAGE]
   Not able to allocate storage for the requested operation.

[GSK_ERROR_IO]
   An error occurred in SSL processing, check the errno value.

Error Conditions
When the gsk_attribute_get_cert_info() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]
   Interrupted function call.

[EDEADLK]
   Resource deadlock avoided.

[ETERM]
   Operation terminated.

If an errno is returned that is not in this list, look in “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Usage Notes
1. After gsk_attribute_get_cert_info() returns with a GSK_OK return value, certDataElem points to an array of structures of type gsk_cert_data_elem. The following structure is the gsk_cert_data_elem structure:

   typedef struct gsk_cert_data_elem_t
   {
      GSK_CERT_DATA_ID cert_data_id;
      char *cert_data_p;
      int cert_data_l;
   } gsk_cert_data_elem;

Each element consists of the following fields:

   - cert_data_id is the identifier for each element of the certificate. The following are the valid identifiers:
     - CERT_BODY_DER (600)
     - CERT_BODY_BASE64 (601)
     - CERT_SERIAL_NUMBER (602)
     - CERT_COMMON_NAME (610)
     - CERT_LOCALITY (611)
     - CERT_STATE_OR_PROVINCE (612)
     - CERT_COUNTRY (613)
     - CERT_ORG (614)
     - CERT_ORG_UNIT (615)
     - CERT_DN_PRINTABLE (616)
     - CERT_DN_DER (617)
     - CERT_POSTAL_CODE (618)
     - CERT_EMAIL (619)
     - CERT_ISSUER_COMMON_NAME (650)
     - CERT_ISSUER_LOCALITY (651)
     - CERT_ISSUER_STATE_OR_PROVINCE (652)
     - CERT_ISSUER_COUNTRY (653)
CERT_ISSUER_ORG (654)
CERT_ISSUER_ORG_UNIT (655)
CERT_ISSUER_DN_PRINTABLE (656)
CERT_ISSUER_DN_DER (657)
CERT_ISSUER_POSTAL_CODE (658)
CERT_ISSUER_EMAIL (659)
CERT_VERSION (660)
CERT_VALID_FROM (662)
CERT_VALID_TO (663)
CERT_PUBLIC_KEY_ALGORITHM (664)
CERT_ISSUER_UNIQUEID (669)
CERT_SUBJECT_UNIQUEID (670)

- cert_data_p points to the specific certificate data.
- cert_data_l contains the length of the data element.

2. Many fields are character strings and are terminated with a trailing null. The length does not include the null.

3. Other fields (CERT_BODY_DER, CERT_DN_DER, and so on) may have imbedded nulls and therefore must use the integer length for processing.

4. Not all certificates contain all fields, so the number of fields returned depends on the certificate being processed. This open-ended approach means new fields can be added from time to time without disrupting existing usage.

5. All certificate data is returned in ASCII CCSID 850.

6. You can reference the certDataElem pointers as long as the handle for the secure session or SSL environment is open.

Related Information

- "gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment" on page 3—Get character information about a secure session or a SSL environment
- "gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment" on page 10—Get enumerated information about a secure session or an SSL environment.
- "gsk_attribute_get_numeric_value()—Get numeric information about a secure session or an SSL environment" on page 14—Get numeric information about a secure session or an SSL environment
- "gsk_environment_close()—Close an SSL environment" on page 30—Close the SSL environment
- "gsk_environment_init()—Initialize an SSL environment" on page 32—Initialize an SSL environment
- "gsk_environment_open()—Get a handle for an SSL environment" on page 34—Get a handle for an SSL environment
- "gsk_secure_soc_close()—Close a secure session" on page 36—Close a secure session
- "gsk_secure_soc_init()—Negotiate a secure session" on page 37—Negotiate a secure session
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session" on page 40—Perform miscellaneous functions for a secure session
- "gsk_secure_soc_open()—Get a handle for a secure session" on page 43—Get a handle for a secure session
- "gsk_strerror()—Retrieve GSKit runtime error message" on page 62—Retrieve GSKit runtime error message
gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment

Syntax
#include <gskssl.h>

int gsk_attribute_get_enum(gsk_handle my_gsk_handle,
                          GSK_ENUM_ID enumID,
                          GSK_ENUM_VALUE *enumValue);

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

The gsk_attribute_get_enum() function is used to obtain values for specific enumerated data for a secure session or an SSL environment.

Parameters

my_gsk_handle (Input)
Indicates one of the following handles:
  • The handle for the secure session. (my_session_handle)
  • The handle for the SSL environment. (my_env_handle)

enumID (Input)
The following values can be used to retrieve information about the secure session or SSL environment that is either defaulted or explicitly set:

  • GSK_PROTOCOL_SSLV2 (403) - Whether the SSL Version 2 protocol is enabled or disabled for this secure session or SSL environment. The enumValue returned will be one of the following values:
    – GSK_PROTOCOL_SSLV2_ON (510) - SSL Version 2 ciphers are enabled.
    – GSK_PROTOCOL_SSLV2_OFF (511) - SSL Version 2 ciphers are disabled.
  • GSK_PROTOCOL_SSLV3 (404) - Whether the SSL Version 3 protocol is enabled or disabled for this secure session or SSL environment. The enumValue returned will be one of the following values:
    – GSK_PROTOCOL_SSLV3_ON (512) - SSL Version 3 ciphers are enabled.
    – GSK_PROTOCOL_SSLV3_OFF (513) - SSL Version 3 ciphers are disabled.
  • GSK_PROTOCOL_TLSV1 (407) - Whether the TLS Version 1 protocol is enabled or disabled for this secure session or SSL environment. The enumValue returned will be one of the following values:
    – GSK_PROTOCOL_TLSV1_ON (518) - TLS Version 1 ciphers are enabled.
    – GSK_PROTOCOL_TLSV1_OFF (519) - TLS Version 1 ciphers are disabled.
  • GSK_SESSION_TYPE (402) - Type of handshake to be used for this secure session or SSL environment. enumValue returned will be one of the following values:
    – GSK_CLIENT_SESSION (507) - Secure sessions act as clients.
    – GSK_SERVER_SESSION (508) - Secure sessions act as a server with no client authentication. The client certificate is not requested.
- **GSK_SERVER_SESSION_WITH_CL_AUTH (509)** - Secure sessions act as a server that requests the client to send a certificate. The value for **GSK_CLIENT_AUTH_TYPE** will determine what happens if the client certificate is not valid or not provided.

- **GSK_CLIENT_AUTH_TYPE (401)** - Type of client authentication to use for this session. `enumValue` must specify one of the following:
  - **GSK_CLIENT_AUTH_FULL (503)** - All received certificates are validated. If a certificate that is not valid is received, the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. If no certificate is sent by the client, the start of the secure session is successful. Applications can detect this situation by checking the GSK_CERTIFICATE_VALIDATION_CODE `enumId` via `gsk_attribute_get_numeric value()`. A `numValue` of GSK_ERROR_NO_CERTIFICATE will indicate no certificate was sent by client. In this case, the application is responsible for the authentication of the client.
  - **GSK_CLIENT_AUTH_PASSTHRU (505)** - All received certificates are validated. If validation is successful or validation fails because the certificate is expired, or does not have a trusted root, the secure session will start. For the other validation failure cases the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. Applications can detect the situation where the secure session started but validation failed by checking the GSK_CERTIFICATE_VALIDATION_CODE `enumId` via `gsk_attribute_get_numeric value()`. The `numValue` will indicate the certificate validation return code for client’s certificate. In this situation, the application is responsible for the authentication of the client.
  - **GSK_OS400_CLIENT_AUTH_REQUIRED (6995)** - All received certificates are validated. If a certificate that is not valid is received, the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. If no certificate is sent by the client, the secure session does not start, and an error code of GSK_ERROR_NO_CERTIFICATE is returned from `gsk_secure_soc_init()`.

- **GSK_PROTOCOL_USED (405)** - Which protocol was used for this secure session. The `enumValue` returned will be one of the following values:
  - **GSK_PROTOCOL_USED_SSLV2 (514)** - The protocol used for this secure session is SSL Version 2.
  - **GSK_PROTOCOL_USED_SSLV3 (515)** - The protocol used for this secure session is SSL Version 3.
  - **GSK_PROTOCOL_USED_TLSV1 (520)** - The protocol used for this secure session is TLS Version 1.

- **GSK_SID_FIRST (406)** - Whether a full handshake or abbreviated handshake occurred for this secure session. The `enumValue` returned will be one of the following values:
  - **GSK_SID_IS_FIRST (516)** - A full handshake occurred for this secure session.
  - **GSK_SID_NOT_FIRST (517)** - An abbreviated handshake occurred for this secure session.

- **GSK_SERVER_AUTH_TYPE (410)** - Type of server authentication to use for this session. `enumValue` must specify one of the following:
  - **GSK_SERVER_AUTH_FULL (534)** - All received certificates are validated. If a certificate that is not valid is received, the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. If no certificate is sent by the server, the secure session does not start, and an error code of GSK_ERROR_NO_CERTIFICATE is returned from `gsk_secure_soc_init()`.

Secure sockets APIs 11
- **GSK_SERVER_AUTH_PASSTHRU (535)** - All received certificates are validated. If validation is successful or validation fails because the certificate has expired or does not have a trusted root, the secure session will start. For the other validation failure cases the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. Applications can detect the situation where the secure session started but validation failed by checking the `GSK_CERTIFICATE_VALIDATION_CODE enumId` via `gsk_attribute_get_numeric_value()`. The `numValue` will indicate the certificate validation return code for server’s certificate. In this situation, the application is responsible for the authentication of the server.

  It is highly recommended that this option only be used if an alternate authentication method is used.

- **GSK_ENVIRONMENT_CLOSE_OPTIONS (411)** - Type of special close options to use for this environment. If `gsk_environment_close()` is issued prior to all secure sessions being closed, the active secure sessions will continue to work and the environment close will effectively be delayed. The resources for the SSL environment will not be freed up until after the last secure session closes. No new secure sessions will be allowed to start using the closed SSL environment. `enumValue` must specify one of the following:
  - **GSK_DELAYED_ENVIRONMENT_CLOSE (536)** - Enable the environment close callback routine support.
  - **GSK_NORMAL_ENVIRONMENT_CLOSE (537)** - Field is ignored.

**enumValue (Output)**

  Specifies a pointer to an integer in which to place the value of the requested information.

**Authorities**

  No authorization is required.

**Return Value**

  `gsk_attribute_get_enum()` returns an integer. Possible values are:

  **[GSK_OK]**

  `gsk_attribute_get_enum()` was successful.

  **[GSK_ATTRIBUTE_INVALID_ID]**

  The specified `enumId` was not valid.

  **[GSK_INVALID_HANDLE]**

  The specified handle was not valid.

  **[GSK_AS400_ERROR_INVALID_POINTER]**

  The `enumValue` pointer is not valid.

  **[GSK_ERROR_UNSUPPORTED]**

  The `enumId` is currently not supported.

  **[GSK_ERROR_IO]**

  An error occurred in SSL processing, check the `errno` value.

**Error Conditions**

  When the `gsk_attribute_get_enum()` API fails with return code `GSK_ERROR_IO`, `errno` can be set to:

  **[EINTR]**

  Interrupted function call.

  **[EDEADLK]**

  Resource deadlock avoided.
Operation terminated.

If an errno is returned that is not in this list, look in “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Usage Notes
1. The following GSK_ENUM_ID values may be retrieved from the SSL environment after gsk_environment_open():
   - GSK_PROTOCOL_SSLV2
   - GSK_PROTOCOL_SSLV3
   - GSK_PROTOCOL_TLSV1
   - GSK_SESSION_TYPE
   - GSK_CLIENT_AUTH_TYPE
   - GSK_SERVER_AUTH_TYPE
   - GSK_ENVIRONMENT_CLOSE_OPTIONS

2. The following GSK_ENUM_ID values may be retrieved from the secure session after gsk_secure_soc_open():
   - GSK_PROTOCOL_SSLV2
   - GSK_PROTOCOL_SSLV3
   - GSK_PROTOCOL_TLSV1
   - GSK_PROTOCOL_USED
   - GSK_SESSION_TYPE
   - GSK_CLIENT_AUTH_TYPE
   - GSK_SID_FIRST
   - GSK_SERVER_AUTH_TYPE

3. The following GSK_ENUM_ID values are defaulted after gsk_secure_soc_open() and will be set for the secure session after gsk_secure_soc_init() or gsk_secure_soc_misc():
   - GSK_PROTOCOL_USED
   - GSK_SID_FIRST

Related Information
- “gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment” on page 3
- “gsk_attribute_get_numeric_value()—Get numeric information about a secure session or an SSL environment” on page 14
- “gsk_attribute_get_cert_info()—Get information about a local or partner certificate” on page 6
- “gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment” on page 23
- “gsk_environment_close()—Close an SSL environment” on page 30
- “gsk_environment_init()—Initialize an SSL environment” on page 32
- “gsk_environment_open()—Get a handle for an SSL environment” on page 34
- “gsk_secure_soc_close()—Close a secure session” on page 36
- “gsk_secure_soc_init()—Negotiate a secure session” on page 37
- “gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40
- “gsk_secure_soc_open()—Get a handle for a secure session” on page 43
- “gsk_strerror()—Retrieve GSKit runtime error message” on page 62
API introduced: V5R1

---

**gsk_attribute_get_numeric_value()**—Get numeric information about a secure session or an SSL environment

Syntax
```
#include <gskssl.h>

int gsk_attribute_get_numeric_value(gsk_handle my_gsk_handle,
                                   GSK_NUM_ID numID,
                                   int *numValue);
```

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

The **gsk_attribute_get_numeric_value()** function is used to obtain specific numeric information about a secure session or an SSL environment.

**Parameters**

**my_gsk_handle (Input)**
Indicates one of the following handles:
- The handle for the secure session. *(my_session_handle)*
- The handle for the SSL environment. *(my_env_handle)*

**numID (Input)**
The following values can be used to retrieve information about the secure session or the SSL environment that is either defaulted or explicitly set:

- **GSK_FD (300)** - *numValue* is a socket descriptor to be used for this secure session.
- **GSK_V2_SESSION_TIMEOUT (301)** - SSL Version 2 session time-out for the environment. *numValue* must be in the range 0-100 seconds.
- **GSK_V3_SESSION_TIMEOUT (302)** - SSL Version 3 and TLS version 1 session time-out for the environment. *numValue* must be in the range 0-86400 seconds.
- **GSK_OS400_READ_TIMEOUT (6993)** - The receive time-out for the secure session or the SSL environment.
- **GSK_CERTIFICATE_VALIDATION_CODE (6996)** - The certificate validation return code for the local or peer certificate.
- **GSK_HANDSHAKE_TIMEOUT (6998)** - SSL handshake time-out for the secure session or the SSL environment.

**numValue (Output)**
A pointer to an integer containing the value of the requested information.

**Authorities**
No authorization is required.
Return Value

gsk_attribute_get_numeric_value() returns an integer. Possible values are:

[GSK_OK]

gsk_attribute_get_numeric_value() was successful.

[GSK_ATTRIBUTE_INVALID_ID]

The specified numID was not valid.

[GSK_INVALID_HANDLE]

The handle specified was not valid.

[GSK_OS400_ERROR_INVALID_POINTER]

The numValue pointer is not valid.

[GSK_ERROR_UNSUPPORTED]

The numID is currently not supported.

[GSK_ERROR_IO]

An error occurred in SSL processing, check the errno value.

Error Conditions

When the gsk_attribute_get_numeric_value() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]

Interrupted function call.

[EDEADLK]

Resource deadlock avoided.

[ETERM]

Operation terminated.

If an errno is returned that is not in this list, look in “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Usage Notes

1. The following GSK_NUM_ID values may be retrieved from the SSL environment after gsk_environment_open():
   - GSK_V2_SESSION_TIMEOUT
   - GSK_V3_SESSION_TIMEOUT
   - GSK_HANDSHAKE_TIMEOUT
   - GSK_OS400_READ_TIMEOUT

2. The following GSK_NUM_ID value may be retrieved from the SSL environment after gsk_environment_init():
   - GSK_CERTIFICATE_VALIDATION_CODE - Will return the certificate validation return code for the local certificate.

3. The following GSK_NUM_ID value may be retrieved from each individual secure session after gsk_secure_soc_init():
   - GSK_CERTIFICATE_VALIDATION_CODE - Will return the certificate validation return code for the peer’s certificate.

4. The following GSK_NUM_ID values may be retrieved from each individual secure session after gsk_secure_soc_open().
The following GSK_NUM_ID values are currently not supported in the i5/OS implementation:

- GSK_V2_SIDCACHE_SIZE
- GSK_V3_SIDCACHE_SIZE
- GSK_LDAP_SERVER_PORT

**Related Information**

- "gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment" on page 3—Get character information about a secure session or an SSL environment.
- "gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment" on page 10—Get enumerated information about a secure session or an SSL environment.
- "gsk_attribute_get_cert_info()—Get information about a local or partner certificate" on page 6—Get information about a local or partner certificate.
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment" on page 28—Set numeric information for a secure session or an SSL environment.
- "gsk_environment_init()—Initialize an SSL environment" on page 32—Initialize an SSL environment.
- "gsk_environment_open()—Get a handle for an SSL environment" on page 34—Get a handle for an SSL environment.
- "gsk_secure_soc_init()—Negotiate a secure session" on page 37—Negotiate a secure session.
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session" on page 40—Perform miscellaneous functions for a secure session.
- "gsk_secure_soc_open()—Get a handle for a secure session" on page 43—Get a handle for a secure session.
- "gsk_strerror()—Retrieve GSK runtime error message" on page 62—Retrieve GSK runtime error message.

API introduced: V5R1

---

**gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment**

**Syntax**

```c
#include <gskssl.h>

int gsk_attribute_set_buffer(gsk_handle my_gsk_handle,
        GSK_BUF_ID bufID,
        const char *buffer,
        int bufSize);
```

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

The **gsk_attribute_set_buffer()** function is used to set a specified buffer attribute to a value inside the specified secure session or SSL environment.
Parameters

**my_gsk_handle (Input)**
Indicates one of the following handles:
- The handle for the secure session. (*my_session_handle*)
- The handle for the SSL environment. (*my_env_handle*)

**bufID (Input)**
Indicates one of the following operations:

- **GSK_KEYRING_FILE (201)** - *buffer* points to the name of the certificate store file to be used for the secure session or SSL environment. Authority to the certificate store file will be checked on the `gsk_environment_init()` API or the `gsk_secure_soc_init()` API.
- **GSK_KEYRING_PW (202)** - *buffer* points to the password for the certificate store file to be used for the secure session or SSL environment.
- **GSK_KEYRING_LABEL (203)** - *buffer* points to the certificate label associated with the certificate in the certificate store to be used for the secure session or SSL environment.
- **GSK_OS400_APPLICATION_ID (6999)** - *buffer* points to the application identifier to be used for the SSL environment.
- **GSK_V2_CIPHER_SPECS (205)** - *buffer* points to the list of SSL Version 2 ciphers to be used for the secure session or the SSL environment.
- **GSK_V3_CIPHER_SPECS (206)** - *buffer* points to the list of SSL Version 3/TLS Version 1 ciphers to be used for the secure session or the SSL environment.

**buffer (Input)**
A pointer to the information to be used for the secure session or the SSL environment.

The data in the buffer is assumed to be represented in the CCSID (coded character set identifier) currently in effect for the job. If the CCSID of the job is 65535, this buffer is assumed to be represented in the default CCSID of the job.

**bufSize (Input)**
The length of the *buffer* information. If *bufSize* is specified as 0, the length of *bufSize* will be calculated.

Authorities
No authorization is required.

Return Value
`gsk_attribute_set_buffer()` returns an integer. Possible values are:

- **[GSK_OK]**
  `gsk_attribute_set_buffer()` was successful.
- **[GSK_ATTRIBUTE_INVALID_ID]**
  The *bufID* value is not a valid identifier.
- **[GSK_ATTRIBUTE_INVALID_LENGTH]**
  The *bufSize* specified or the length of *buffer* is not valid.
- **[GSK_INVALID_HANDLE]**
  *my_gsk_handle* is not a valid handle that was received from issuing `gsk_environment_open()` or `gsk_secure_soc_open()`.
- **[GSK_AS400_ERROR_INVALID_POINTER]**
  The *buffer* pointer is not valid.
[GSK_INVALID_STATE]
One of the following occurred:
- bufID cannot be set for a SSL environment after a `gsk_environment_init()` has been issued.
- bufID cannot be set for a secure session after a `gsk_secure_soc_init()` has been issued.

[GSK_ERROR_UNSUPPORTED]
The bufID value is currently not supported.

[GSK_INSUFFICIENT_STORAGE]
Not able to allocate storage for the requested operation.

[GSK_ERROR_IO]
An error occurred in SSL processing, check the `errno` value.

Error Conditions
When the `gsk_attribute_set_buffer()` API fails with return code [GSK_ERROR_IO], `errno` can be set to:

[EINTR]
Interrupted function call.

[EDEADLK]
Resource deadlock avoided.

[ETERM]
Operation terminated.

If an `errno` is returned that is not in this list, look in "Errno Values for UNIX-Type Functions” on page 101 for a description of the `errno`.

Usage Notes
1. The following `GSK_BUF_ID` values may be set in the SSL environment after `gsk_environment_open()` and before `gsk_environment_init()`. They are used as defaults for subsequent secure sessions:

- `GSK_KEYRING_FILE`
- `GSK_KEYRING_PW`
- `GSK_KEYRING_LABEL`
- `GSK_OS400_APPLICATION_ID`
- `GSK_V2_CIPHER_SPECS`
- `GSK_V3_CIPHER_SPECS`

2. The following `GSK_BUF_ID` values may be set for each individual secure session after `gsk_secure_soc_open()` and before `gsk_secure_soc_init()`. These values will override values set in the SSL environment:

- `GSK_KEYRING_LABEL`
- `GSK_V2_CIPHER_SPECS`
- `GSK_V3_CIPHER_SPECS`

3. The following `GSK_V3_CIPHER_SPECS` values are the SSL Version 3 ciphers and the TLS Version 1 ciphers supported:

- 01 = NULL MD5
- 02 = NULL SHA
- 03 = RC4 MD5 EXPORT
- 04 = RC4 MD5 US
- 05 = RC4 SHA US
- 06 = RC2 MD5 EXPORT
09 = DES SHA EXPORT
0A = Triple DES SHA US

2F = TLS_RSA_WITH_AES_128_CBC_SHA
35 = TLS_RSA_WITH_AES_256_CBC_SHA
NULL = Default cipher specs are used (may change in future)

The default list is '04052F350A090306'

4. The following `GSK_V2_CIPHER_SPECS` values are the SSL Version 2 ciphers supported:

1 = RC4 US
2 = RC4 EXPORT
3 = RC2 US
4 = RC2 EXPORT
6 = DES 56-bit
7 = Triple DES US
NULL = Default cipher specs are used (may change in future)

The default list is '136724'

5. The following `GSK_BUF_ID` values currently are not supported in the i5/OS implementation:

- `GSK_KEYRING_STASH_FILE`
- `GSK_LDAP_SERVER`
- `GSK_LDAP_USER`
- `GSK_LDAP_USER_PW`
- `GSK_USER_DATA`
- `GSK_SID_VALUE`
- `GSK_PKCS11_DRIVER_PATH`
- `GSK_PKCS11_TOKEN_LABEL`
- `GSK_PKCS11_TOKEN_PWD`
- `GSK_CSP_NAME`

6. The following are the possible scenarios for the use of `GSK_KEYRING_LABEL`:

- `GSK_KEYRING_LABEL` can be set after `gsk_environment_open()` and before `gsk_environment_init()` to indicate which certificate in the `GSK_KEYRING_FILE` to use for the secure environment.
- `GSK_KEYRING_LABEL` can be set after `gsk_secure_soc_open()` and before `gsk_secure_soc_init()` to indicate which certificate in the `GSK_KEYRING_FILE` to use for the secure session.
- If `GSK_KEYRING_LABEL` is not set, the default certificate label in the `GSK_KEYRING_FILE` is used for the SSL environment.

7. If `GSK_OS400_APPLICATION_ID` is set, the `GSK_KEYRING_FILE`, the `GSK_KEYRING_LABEL`, and the `GSK_KEYRING_PASSWORD` values are ignored.

Related Information

- "gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment" on page 3
- "gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment" on page 23
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment" on page 28
The `gsk_attribute_set_callback()` function is used to set callback pointers to routines in the user application. These routines may be used for special purposes by the application.

### Parameters

**my_gsk_handle (Input)**

Indicates one of the following handles:
- The handle for the secure session. (`my_session_handle`)
- The handle for the SSL environment. (`my_env_handle`)

**callBackID (Input)**

Indicates one of the following operations:

- **GSK_ENVIRONMENT_CLOSE_CALLBACK (804)** - This is the callback to have a user routine be called when the last secure session is closed that was created based on secure environment that already has been closed.
- **GSK_CERT_VALIDATION_CALLBACK (805)** - This is the callback that is required to do additional certificate validation

**callBackAreaPtr (Input)**

Address of a callback routine or address of a structure containing pointers to callback routines appropriate to the `callBackID`. The following indicate what should be pointed to by the `callBackAreaPtr` based on the value of the `callBackID`.

- **GSK_ENVIRONMENT_CLOSE_CALLBACK** - The `callBackAreaPtr` should be set to the address of a function with prototype `pGSK_ENVIRONMENT_CLOSE_CALLBACK`. `pGSK_ENVIRONMENT_CLOSE_CALLBACK` is defined as:
  ```c
  typedef void (*pGSK_ENVIRONMENT_CLOSE_CALLBACK)(gsk_handle my_env_handle);
  ```

---

**API introduced:** V5R1
• **GSK_CERT_VALIDATION_CALLBACK** - The `callBackAreaPtr` must point to a `validationCallBack` structure. That structure is defined as:

```c
typedef struct validationCallBack_struct
{
    pgsk_cert_validation_callback validation_callback;
    VALIDATE_REQUIRED validateRequired;
    CERT_NEEDED certificateNeeded;
} validationCallBack;
```

- `validationCallBack.validation_callback` should be set to a value of type `pgsk_cert_validation_callback`. This is the `pgsk_cert_validation_callback` typedef:

```c
typedef int (*pgsk_cert_validation_callback)(const unsigned char * my_CertificateChain, int my_validation_status);
```

Do not use `pgsk_cert_validation_callback` as a variable type when you create your prototype and function though. The following prototype should be used in the code for the function whose address will be assigned to `validationCallBack.validation_callback`:

```c
int foo(const unsigned char * my_CertificateChain, int my_validation_status);
```

The return value from this function will be one of following:

- **GSK_OK (0)** - Application accepts the certificate, and SSL will continue the handshake with this value.
- **GSK_ERROR_CERT_VALIDATION (8)** - Application does not accept the certificate, and SSL handshake will terminate immediately with this value. If callback routine return other than **GSK_OK**, SSL will consider it as **GSK_ERROR_CERT_VALIDATION** and terminate the handshake.

**Parameters**

**my_CertificateChain (Input)**

A pointer to a copy of buffer which contains the data of certificate chain.

**my_validation_status (Input)**

Results from SSL certificate validation:

- **GSK_VALIDATION_SUCCESSFUL (0)** - Validation is successful.
- **GSK_OS400_ERROR_NOT_TRUSTED_ROOT (6000)** - The certificate is not signed by a trusted certificate authority.
- **GSK_KEYFILE_CERT_EXPIRED (107)** - The validity time period of the certificate has expired.

- `validationCallBack.validateRequired` - This is the flag to inform SSL when to call the certificate validation callback. The following values can be used:

- **GSK_NO_VALIDATION (900)** - User application would like SSL to validate and authenticate the certificate first before calling the certificate validation callback. However, if validation fails because the certificate is expired or does not have a trusted root the certificate validation callback will still be called.
- **GSK_VALIDATION_REQUIRED (901)** - User application would like SSL to validate and authenticate the certificate first before calling the certificate validation callback.

- NOTE: If Authentication PassThru is set, and the application set the certificate callback to **GSK_VALIDATION_REQUIRED**, SSL will reject the call with an error code **GSK_CONFLICTING_VALIDATION_SETTING**. If a certificate validation callback has been set to **GSK_VALIDATION_REQUIRED**, and application set authentication to
PassThru, SSL will also reject the call with an error code GSK_CONFLICTING_VALIDATION_SETTING.

- validationCallBack.certificateNeeded - Provides certificate chain flag which informs SSL what certificate chain should be passed to the certificate validation callback. The following values can be used:

- GSK_COMPLETED_CERTIFICATE_CHAIN (951) - To pass the callback routine the complete certificate chain built by SSL during certificate validation and authentication.
- GSK_CERTIFICATE_CHAIN_SENT_VIA_SSL (950) - To pass the callback routine the complete certificate chain built by SSL during certificate validation and authentication.
- GSK_END_ENTITY_CERTIFICATE (952) - To pass the callback routine the EE certificate only. Note: This value will be ignored when the user set certificate validation flag to GSK_NO_VALIDATION. In other words, SSL will set it to GSK_CERTIFICATE_CHAIN_SENT_VIA_SSL.

Authorities
No authorization is required.

Return Value
gsk_attribute_set_callback() returns an integer. Possible values are:

[GSK_OK]
gsk_attribute_set_callback() was successful.

[GSK_ATTRIBUTE_INVALID_ID]
The callBackID specified was not valid.

[GSK_ATTRIBUTE_INVALID_ENUMERATION]
An enumeration referenced by the callBackAreaPtr was not valid.

[GSK_CONFLICTING_VALIDATION_SETTING]
The value for the validationCallBack.validateRequired field for GSK_CERTIFICATION_CALLBACK conflicts with the setting for either GSK_SERVER_AUTH_TYPE or GSK_CLIENT_AUTH_TYPE set by gsk_attribute_set_enum().

[GSK_INVALID_STATE]
The callBackID cannot be set after a gsk_environment_init() has been issued.

[GSK_INVALID_HANDLE]
The handle specified was not valid.

[GSK_ERROR_UNSUPPORTED]
The callBackID is currently not supported.

[GSK_ERROR_IO]
An error occurred in SSL processing, check the errno value.

Error Conditions
When the gsk_attribute_set_callback() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]
Interrupted function call.

[EDEADLK]
Resource deadlock avoided.
Operation terminated.

If an errno is returned that is not in this list, look in “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Usage Notes

1. The following GSK_CALLBACK_ID values may be set in the SSL environment after gsk_environment_open() and before gsk_environment_init(). They are used as defaults for subsequent secure sessions:
   - GSK_ENVIRONMENT_CLOSE_CALLBACK
   - GSK_CERT_VALIDATION_CALLBACK

2. The following GSK_CALLBACK_ID values currently are not supported in the i5/OS implementation:
   - GSK_IO_CALLBACK
   - GSK_SID_CACHE_CALLBACK
   - GSK_CLIENT_CERT_CALLBACK
   - GSK_PKCS11_CALLBACK

Related Information

- "gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment"—Set enumerated information for a secure session or an SSL environment.
- "gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment” on page 16—Set character string information for a secure session or an SSL environment.
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment” on page 28—Set numeric information for a secure session or an SSL environment.
- "gsk_environment_init()—Initialize an SSL environment” on page 32—Initialize an SSL environment.
- "gsk_environment_open()—Get a handle for an SSL environment” on page 34—Get a handle for an SSL environment.
- "gsk_secure_soc_init()—Negotiate a secure session” on page 37—Negotiate a secure session.
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40—Perform miscellaneous functions for a secure session.
- "gsk_secure_soc_open()—Get a handle for a secure session” on page 43—Get a handle for a secure session.
- "gsk_strerror()—Retrieve GSKit runtime error message” on page 62—Retrieve GSK runtime error message.

API introduced: V5R3

---

**gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment**

Syntax


```c
#include <gskssl.h>

int gsk_attribute_set_enum(gsk_handle my_gsk_handle,
    GSK_ENUM_ID enumID,
    GSK_ENUM_VALUE enumValue);
```

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

The `gsk_attribute_set_enum()` function is used to set a specified enumerated type attribute to an enumerated value in the secure session or SSL environment.

**Parameters**

**my_gsk_handle** (Input)
- Indicates one of the following handles:
  - The handle for the secure session. `{my_session_handle}`
  - The handle for the SSL environment. `{my_env_handle}`

**enumID** (Input)
- Indicates one of the following operations:
  - `GSK_PROTOCOL_SSLV2` (403) - Enables or disables the SSL Version 2 protocol. `enumValue` must specify one of the following:
    - `GSK_PROTOCOL_SSLV2_ON` (510) - Enable SSL Version 2 ciphers.
    - `GSK_PROTOCOL_SSLV2_OFF` (511) - Disable SSL Version 2 ciphers.
  - `GSK_PROTOCOL_SSLV3` (404) - Enables or disables the SSL Version 3 protocol. `enumValue` must specify one of the following:
    - `GSK_PROTOCOL_SSLV3_ON` (512) - Enable SSL Version 3 ciphers.
    - `GSK_PROTOCOL_SSLV3_OFF` (513) - Disable SSL Version 3 ciphers.
  - `GSK_PROTOCOL_TLSV1` (407) - Enables or disables the TLS Version 1 protocol. `enumValue` must specify one of the following:
    - `GSK_PROTOCOL_TLSV1_ON` (518) - Enable TLS Version 1 ciphers.
    - `GSK_PROTOCOL_TLSV1_OFF` (519) - Disable TLS Version 1 ciphers.
  - `GSK_SESSION_TYPE` (402) - Type of handshake to be used for this secure session or SSL environment. `enumValue` must specify one of the following operations:
    - `GSK_CLIENT_SESSION` (507) - Secure sessions act as clients.
    - `GSK_SERVER_SESSION` (508) - Secure sessions act as a server with no client authentication. The client is not asked for a certificate.
    - `GSK_SERVER_SESSION_WITH_CL_AUTH` (509) - Secure sessions act as a server that requests the client to send a certificate. The value for `GSK_CLIENT_AUTH_TYPE` will determine what happens if the client certificate is not valid or not provided.
  - `GSK_CLIENT_AUTH_TYPE` (401) - Type of client authentication to use for this session. `enumValue` must specify one of the following:
    - `GSK_CLIENT_AUTH_FULL` (503) - All received certificates are validated. If an invalid certificate is received, the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`.

If no certificate is sent by the client, the start of the secure session is successful. Applications can detect this situation by checking the `GSK_CERTIFICATE_VALIDATION_CODE` `enumId`
through `gsk_attribute_get_numeric value()`. A `numValue` of GSK_ERROR_NO_CERTIFICATE will indicate no certificate was sent by client. In this case, the application is responsible for the authentication of the client.

- **GSK_CLIENT_AUTH_PASSTHRU (505)** - All received certificates are validated. If validation is successful or validation fails because the certificate is expired or does not have a trusted root, the secure session will start. For the other validation failure cases the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. Applications can detect the situation where the secure session started but validation failed by checking the GSK_CERTIFICATE_VALIDATION_CODE `enumId` via `gsk_attribute_get_numeric value()`. The `numValue` will indicate the certificate validation return code for client’s certificate. In this situation, the application is responsible for the authentication of the client.

If no certificate is sent by the client, the start of the secure session is successful. Applications can detect this situation by checking the GSK_CERTIFICATE_VALIDATION_CODE `enumId` as well. A `numValue` of GSK_ERROR_NO_CERTIFICATE will indicate no certificate was sent by client. In this case, the application is also responsible for the authentication of the client.

NOTE: If Authentication PassThru is set, and the application set the certificate callback to GSK_VALIDATION_REQUIRED, SSL will reject the call with an error code GSK_CONFLICTING_VALIDATION_SETTING. If a certificate validation callback has been set to GSK_VALIDATION_REQUIRED, and application set authentication to PassThru, SSL will also reject the call with an error code GSK_CONFLICTING_VALIDATION_SETTING.

- **GSK_OS400_CLIENT_AUTH_REQUIRED (6995)** - All received certificates are validated. If a certificate that is not valid is received, the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. If no certificate is sent by the client, the secure session does not start, and an error code of GSK_ERROR_NO_CERTIFICATE is returned from `gsk_secure_soc_init()`.

- **GSK_SERVER_AUTH_TYPE (410)** - Type of server authentication to use for this session. `enumValue` must specify one of the following:

  - **GSK_SERVER_AUTH_FULL (534)** - All received certificates are validated. If a certificate that is not valid is received, the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. If no certificate is sent by the server, the secure session does not start, and an error code of GSK_ERROR_NO_CERTIFICATE is returned from `gsk_secure_soc_init()`.

  - **GSK_SERVER_AUTH_PASSTHRU (535)** - All received certificates are validated. If validation is successful or validation fails because the certificate has expired or does not have a trusted root, the secure session will start. For the other validation failure cases the secure session does not start, and an error code is returned from `gsk_secure_soc_init()`. Applications can detect the situation where the secure session started but validation failed by checking the GSK_CERTIFICATE_VALIDATION_CODE `enumId` via `gsk_attribute_get_numeric value()`. The `numValue` will indicate the certificate validation return code for server’s certificate. In this situation, the application is responsible for the authentication of the server.

It is highly recommended that this option only be used if an alternate authentication method is used.

NOTE: If Authentication PassThru is set, and the application set the certificate callback to GSK_VALIDATION_REQUIRED, SSL will reject the call with an error code GSK_CONFLICTING_VALIDATION_SETTING. If a certificate validation callback has been set to GSK_VALIDATION_REQUIRED, and application set authentication to PassThru, SSL will also reject the call with an error code GSK_CONFLICTING_VALIDATION_SETTING.

- **GSK_ENVIRONMENT_CLOSE_OPTIONS (411)** - Type of special close options to use for this environment. If `gsk_environment_close()` is issued prior to all secure sessions being closed, the active secure sessions will continue to work and the environment close will effectively be delayed. The resources for the SSL environment will not be freed up until after the last secure...
session closes. No new secure sessions will be allowed to start using the closed SSL environment. *enumValue* must specify one of the following:

- **GSK_DELAYED_ENVIRONMENT_CLOSE (536)** - Enable the environment close callback routine support.
- **GSK_NORMAL_ENVIRONMENT_CLOSE (537)** - Field is ignored.

*enumValue* (Input)
An enumerated type appropriate to the *enumID*.

**Authorities**
No authorization is required.

**Return Value**
gsk_attribute_set_enum() returns an integer. Possible values are:

- **[GSK_OK]**
gsk_attribute_set_enum() was successful.
- **[GSK_ATTRIBUTE_INVALID_ENUMERATION]**
The enumeration specified for the *enumValue* was not valid.
- **[GSK_ATTRIBUTE_INVALID_ID]**
The *enumID* specified was not valid.
- **[GSK_CONFLICTING_VALIDATION_SETTING]**
The value for GSK_SERVER_AUTH_TYPE or GSK_CLIENT_AUTH_TYPE conflicts with the setting for the validationCallBack.validateRequired field for GSK_CERT_VALIDATION_CALLBACK set by gsk_attribute_set_callback().
- **[GSK_INVALID_STATE]**
One of the following occurred:
  - The *enumID* cannot be set after a gsk_environment_init() has been issued.
  - The *enumID* cannot be set after a gsk_secure_soc_init() has been issued.
- **[GSK_INVALID_HANDLE]**
The handle specified was not valid.
- **[GSK_ERROR_UNSUPPORTED]**
The *enumID* is currently not supported.
- **[GSK_ERROR_IO]**
An error occurred in SSL processing, check the *errno* value.

**Error Conditions**
When the gsk_attribute_set_enum() API fails with return code [GSK_ERROR_IO], *errno* can be set to:

- **[EINTR]**
  Interrupted function call.
- **[EDEADLK]**
  Resource deadlock avoided.
- **[ETERM]**
  Operation terminated.

If an *errno* is returned that is not in this list, look in "Errno Values for UNIX-Type Functions" on page 101 for a description of the *errno*. 
Usage Notes

1. The following GSK_ENUM_ID values may be set in the SSL environment after gsk_environment_open() and before gsk_environment_init(). They are used as defaults for subsequent secure sessions:

- GSK_PROTOCOL_SSLV2
- GSK_PROTOCOL_SSLV3
- GSK_PROTOCOL_TLSV1
- GSK_SESSION_TYPE
- GSK_CLIENT_AUTH_TYPE
- GSK_SERVER_AUTH_TYPE
- GSK_ENVIRONMENT_CLOSE_OPTIONS

2. The following GSK_ENUM_ID values may be set for each individual secure session after gsk_secure_soc_open() and before gsk_secure_soc_init(). These values will override values set in the SSL environment:

- GSK_PROTOCOL_SSLV2
- GSK_PROTOCOL_SSLV3
- GSK_PROTOCOL_TLSV1
- GSK_SESSION_TYPE
- GSK_CLIENT_AUTH_TYPE
- GSK_SERVER_AUTH_TYPE

Related Information

- "gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment" on page 10—Get enumerated information about a secure session or an SSL environment.
- "gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment" on page 16—Set character string information for a secure session or an SSL environment.
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment" on page 28—Set numeric information for a secure session or an SSL environment.
- "gsk_environment_init()—Initialize an SSL environment" on page 32—Initialize an SSL environment.
- "gsk_environment_open()—Get a handle for an SSL environment" on page 34—Get a handle for an SSL environment.
- "gsk_secure_soc_init()—Negotiate a secure session" on page 37—Negotiate a secure session.
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session" on page 40—Perform miscellaneous functions for a secure session.
- "gsk_secure_soc_open()—Get a handle for a secure session" on page 43—Get a handle for a secure session.
- "gsk_strerror()—Retrieve GSkIt runtime error message" on page 62—Retrieve GSK runtime error message.

API introduced: V5R1
The `gsk_attribute_set_numeric_value()` function is used to set specific numeric information for a secure session or an SSL environment.

**Parameters**

**my_gsk_handle** (Input)
- One of the following handles:
  - The handle for the secure session. *(my_session_handle)*
  - The handle for the SSL environment. *(my_env_handle)*

**numID** (Input)
- One of the following operations:
  - GSK_FD (300) - `numValue` is a socket descriptor to be used for this secure session.
  - GSK_V2_SESSION_TIMEOUT (301) - `numValue` is the SSL Version 2 session time-out for the SSL environment. `numValue` must be in the range 0-100 seconds.
  - GSK_V3_SESSION_TIMEOUT (302) - `numValue` is the SSL Version 3 and TLS Version 1 session time-out for the SSL environment. `numValue` must be in the range 0-86400 seconds (24 hours).
  - GSK_OS400_READ_TIMEOUT (6993) - `numValue` is the receive time-out for the secure session or the SSL environment. `numValue` must be in milliseconds. A `numValue` of 0 is the default which means to wait forever.
  - GSK_HANDSHAKE_TIMEOUT (6998) - `numValue` is the SSL handshake time-out for the secure session or the SSL environment. `numValue` must be in seconds. A `numValue` of 0 is the default which means to wait forever.

**numValue** (Input)
- An integer value to be updated for the specified `numID`.

**Authorities**
No authorization is required.

**Return Value**

`gsk_attribute_set_numeric_value()` returns an integer. Possible values are:

[GSK_OK]

`gsk_attribute_set_numeric_value()` was successful.

[GSK_INVALID_STATE]
One of the following occurred:

- `numID` cannot be set in the SSL environment after a `gsk_environment_init()` has been issued.
- `numID` cannot be set for a secure session after a `gsk_secure_soc_init()` has been issued.

[GSK_ATTRIBUTE_INVALID_ID]
The `numID` specified was not valid.

[GSK_ATTRIBUTE_INVALID_NUMERIC_VALUE]
The `numValue` specified was not valid.

[GSK_INVALID_HANDLE]
A handle was specified that was not valid.

[GSK_ERROR_UNSUPPORTED]
The `numID` is currently not supported.

[GSK_ERROR_IO]
An error occurred in SSL processing, check the `errno` value.

**Error Conditions**

When the `gsk_attribute_set_numeric_value()` API fails with return code `[GSK_ERROR_IO]`, `errno` can be set to:

- [EINTR]
  Interrupted function call.

- [EDEADLK]
  Resource deadlock avoided.

- [ETERM]
  Operation terminated.

If an `errno` is returned that is not in this list, look in "Errno Values for UNIX-Type Functions" on page 101 for a description of the `errno`.

**Usage Notes**

1. The following `GSK_NUM_ID` values may be set in the SSL environment after `gsk_environment_open()` and before `gsk_environment_init()`. They are used as defaults for subsequent secure sessions:

   - `GSK_V2_SESSION_TIMEOUT`
   - `GSK_V3_SESSION_TIMEOUT`
   - `GSK_HANDSHAKE_TIMEOUT`
   - `GSK_OS400_READ_TIMEOUT`

2. The following `GSK_NUM_ID` values may be set for each individual secure session after `gsk_secure_soc_open()` and before `gsk_secure_soc_init()`. These values will override values set in the SSL environment:

   - `GSK_FD`
   - `GSK_HANDSHAKE_TIMEOUT`
   - `GSK_OS400_READ_TIMEOUT`

3. The following `GSK_NUM_ID` values are currently not supported in the i5/OS implementation:

   - `GSK_V2_SIDCACHE_SIZE`
• GSK_V3_SIDCACHE_SIZE
• GSK_LDAP_SERVER_PORT

4. The GSK_FD value is a socket descriptor that must have an address family of AF_INET or AF_INET6 and a socket type of SOCK_STREAM.

Related Information
• “gsk_attribute_get_numeric_value()—Get numeric information about a secure session or an SSL environment” on page 14—Get numeric information about a secure session or an SSL environment
• “gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment” on page 16—Set character string information for a secure session or an SSL environment.
• “gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment” on page 23—Set enumerated information for a secure session or an SSL environment.
• “gsk_environment_init()—Initialize an SSL environment” on page 32—Initialize an SSL environment.
• “gsk_environment_open()—Get a handle for an SSL environment” on page 34—Get a handle for an SSL environment.
• “gsk_secure_soc_init()—Negotiate a secure session” on page 37—Negotiate a secure session.
• “gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40—Perform miscellaneous functions for a secure session.
• “gsk_secure_soc_open()—Get a handle for a secure session” on page 43—Get a handle for a secure session.
• “gsk_strerror()—Retrieve GSK runtime error message” on page 62—Retrieve GSK runtime error message.

API introduced: V5R1

---

gsk_environment_close()—Close an SSL environment

Syntax
#include <gskssl.h>

int gsk_environment_close(gsk_handle *my_env_handle);

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

The gsk_environment_close() function is used to close the SSL environment and release all storage associated with the environment.

Parameters
my_env_handle (Input)
A pointer to the handle for the SSL environment to be closed.

Authorities
No authorization is required.
Return Value

gsk_environment_close() returns an integer. Possible values are:

[GSK_OK]

gsk_environment_close() was successful.

[GSK_CLOSE_FAILED]

An error occurred during close processing.

[GSK_INVALID_HANDLE]

The handle specified was not valid.

[GSK_AS400_ERROR_INVALID_POINTER]

my_env_handle pointer is not valid.

[GSK_ERROR_IO]

An error occurred in SSL processing, check the errno value.

Error Conditions

When the gsk_environment_close() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]

Interrupted function call.

[EDeadLK]

Resource deadlock avoided.

[ETERM]

Operation terminated.

If an errno is returned that is not in this list, look in “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Error Messages

Message ID  Error Message Text
CPE3418 E  Possible APAR condition or hardware failure.
CPF9872 E  Program or service program &1 in library &2 ended. Reason code &3.
CPFA081 E  Unable to set return value or error code.

Usage Notes

1. You should close all secure sessions using the SSL environment prior to doing the gsk_environment_close().

2. If gsk_environment_close() is issued prior to all secure sessions being closed, the active secure sessions will continue to work. The resources for the SSL environment will not be freed up until after the last secure session closes. No new secure sessions will be allowed to start using the closed SSL environment.

Related Information

• “gsk_environment_init()—Initialize an SSL environment” on page 32—Initialize an SSL environment

• “gsk_environment_open()—Get a handle for an SSL environment” on page 34—Get a handle for an SSL environment

• “gsk_secure_soc_close()—Close a secure session” on page 36—Close a secure session
gsk_environment_init()—Initialize an SSL environment

Syntax
#include <gskssl.h>

int gsk_environment_init(gsk_handle my_env_handle);

Service Program Name: QSYS/QSOSSLSR
Default Public Authority: *USE
Threading: Yes

The gsk_environment_init() function is used to initialize the SSL environment after any required attributes are set. The certificate store file is opened and other operations such as accessing information in the registration facility are performed to set up this environment. After this function call is issued, SSL is ready to process secure session requests.

Parameters
my_env_handle (Input)
   The handle identifying the SSL environment that will be initialized.

Authorities
Authorization of *R (allow access to the object) to the certificate store file and its associated files is required. Authorization of *X (allow use of the object) to each directory of the path name of the certificate store file and its associated files is required.

Return Value
gsk_environment_init() returns an integer. Possible values are:

[GSK_OK]
   gsk_environment_init() was successful.

[GSK_INVALID_HANDLE]
   The handle specified was not valid.

[GSK_INVALID_STATE]
   A gsk_environment_init() has already been issued with this handle.

[GSK_KEYRING_OPEN_ERROR]
   Certificate store file could not be opened.

[GSK_AS400_ERROR_NO_ACCESS]
   No permission to access the certificate store file.
[GSK_ERROR_BAD_V3_CIPHER]
An SSLv3 or TLSv1 cipher suite was specified that is not valid.

[GSK_ERROR_BAD_V2_CIPHER]
An SSLv2 cipher suite was specified that is not valid.

[GSK_ERROR_BAD_CERTIFICATE]
The certificate is bad.

[GSK_ERROR_NO_PRIVATE_KEY]
There is no private key associated with the certificate.

[GSK_ERROR_BAD_KEYFILE_LABEL]
The specified certificate store's certificate label is not valid or does not exist.

[GSK_ERROR_BAD_KEYFILE_PASSWORD]
The specified certificate store password is not valid.

[GSK_NO_KEYFILE_PASSWORD]
No certificate store password was specified.

[GSK_AS400_ERROR_NOT_REGISTERED]
The application identifier has not been registered.

[GSK_AS400_ERROR_INVALID_POINTER]
my_env_handle pointer is not valid.

[GSK_ERROR_BAD_KEY_LEN_FOR_EXPORT]
The certificate was created with a key length that cannot be exported.

[GSK_INSUFFICIENT_STORAGE]
Not able to allocate storage for the requested operation.

[GSK_INTERNAL_ERROR]
An unexpected error occurred during SSL processing.

[GSK_ERROR_IO]
An error occurred in SSL processing, check errno value.

Error Conditions
When the gsk_environment_init() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]
Interrupted function call.

[EDEADLK]
Resource deadlock avoided.

[ETERM]
Operation terminated.
If an *errno* is returned that is not in this list, look in "Errno Values for UNIX-Type Functions" on page 101 for a description of the *errno*.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

**Usage Notes**

1. If *gsk_environment_init()* fails, *gsk_environment_close()* must be issued to clean up resources.
2. Multiple SSL environment handles may be opened in a process with different attributes set for each SSL environment.
3. The status of the local certificate can be determined by checking the GSK_CERTIFICATE_VALIDATION_CODE enumId using *gsk_attribute_get_numeric_value()*.

**Related Information**

- "gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment" on page 16
- "gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment" on page 23
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment" on page 28
- "gsk_environment_close()—Close an SSL environment" on page 30
- "gsk_environment_open()—Get a handle for an SSL environment"—Get a handle for an SSL environment
- "gsk_strerror()—Retrieve GSKit runtime error message" on page 62

API introduced: V5R1

---

**gsk_environment_open()—Get a handle for an SSL environment**

Syntax

```c
#include <gskssl.h>

int gsk_environment_open(gsk_handle *my_env_handle);
```

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

The *gsk_environment_open()* function is used to get storage for the SSL environment. This function call must be issued before any other gsk function calls are issued. This call returns an SSL environment handle that must be saved and used on subsequent gsk calls.
Parameters

my_env_handle (Output)

A pointer to the SSL environment handle to be used for subsequent gsk function calls.

Authorities

No authorization is required.

Return Value

gsk_environment_open() returns an integer. Possible values are:

[GSK_OK]

gsk_environment_open() was successful.

[GSK_API_NOT_AVAILABLE]

Digital Certificate Manager (DCM), 57xx-SS1 - OS400 Option 34 is not installed.

[GSK_INSUFFICIENT_STORAGE]

Not able to allocate storage for the requested operation.

[GSK_INTERNAL_ERROR]

An internal error occurred during system processing.

[GSK_AS400_ERROR_INVALID_POINTER]

The my_env_handle pointer is not valid.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. After gsk_environment_open() returns with a GSK_OK return value, attributes for the SSL environment have been set and can be retrieved using any of the get function calls. The following is a list of the defaulted values:
   - GSK_V2_SESSION_TIMEOUT set to 100 seconds.
   - GSK_V3_SESSION_TIMEOUT set to 86400 seconds (24 hours).
   - GSK_HANDSHAKE_TIMEOUT set to 0 (wait forever).
   - GSK_OS400_RECEIVE_TIMEOUT set to 0 (wait forever).
   - GSK_SESSION_TYPE set to GSK_CLIENT_SESSION.
   - GSK_KEYRING_LABEL set to use the default certificate from the certificate store file.
   - GSK_PROTOCOL_TLSV1 set to GSK_PROTOCOL_TLSV1_ON.
   - GSK_PROTOCOL_SSLV3 set to GSK_PROTOCOL_SSLV3_ON.
   - GSK_PROTOCOL_SSLV2 set to GSK_PROTOCOL_SSLV2_ON.
   - GSK_V2_CIPHER_SPECS set to the default SSL Version 2 cipher suite list.
   - GSK_V3_CIPHER_SPECS set to the default SSL Version 3 cipher suite list.

2. The default cipher suite list in preference order is as follows:
- **GSK_V3_CIPHER_SPECS** set to SSL Version 3 or TLS Version 1 default "04052F350A090306."
- **GSK_V2_CIPHER_SPECS** set to "137624."

See the usage notes in "gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment" on page 16 API for the format of the ciphers.

**Related Information**
- "gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment" on page 16
- "gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment" on page 23
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment" on page 28
- "gsk_environment_close()—Close an SSL environment" on page 30
- "gsk_environment_init()—Initialize an SSL environment" on page 32
- "gsk_strerror()—Retrieve GSK runtime error message" on page 62

API introduced: V5R1

---

### gsk_secure_soc_close()—Close a secure session

**Syntax**

```
#include <gskssl.h>

int gsk_secure_soc_close(gsk_handle *my_session_handle);
```

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

The **gsk_secure_soc_close()** function is used to close a secure session and free all the associated resources for that secure session.

**Parameters**

- **my_session_handle (Input)**  
  A pointer to the handle for the secure session to be closed. This handle originated from a call to **gsk_secure_soc_open()**.

**Authorities**

No authorization is required.

**Return Value**

- **gsk_secure_soc_close()** returns an integer. Possible values are:
  - [GSK_OK]  
    - **gsk_secure_soc_close()** was successful.
  - [GSK_CLOSE_FAILED]
An error occurred during close processing.

[GSK_INVALID_HANDLE]

The handle specified was not valid.

[GSK_ERROR_IO]

An error occurred in SSL processing, check the _errno_ value.

**Error Conditions**

When the _gsk_secure_soc_close()_ API fails with return code [GSK_ERROR_IO], _errno_ can be set to:

[EINTR]

Interrupted function call.

[EDEADLK]

Resource deadlock avoided.

[ETERM]

Operation terminated.

If an _errno_ is returned that is not in this list, look in “Errno Values for UNIX-Type Functions” on page 101 for a description of the _errno_.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

**Usage Notes**

1. You must do a _gsk_secure_soc_close()_ if a prior _gsk_secure_soc_open()_ was successful.

**Related Information**

- “_gsk_secure_soc_init()_—Negotiate a secure session”—Negotiate a secure session
- “_gsk_secure_soc_misc()_—Perform miscellaneous functions for a secure session” on page 40—Perform miscellaneous functions for a secure session
- “_gsk_secure_soc_open()_—Get a handle for a secure session” on page 43—Get a handle for a secure session
- “_gsk_strerror()_—Retrieve GSKit runtime error message” on page 62—Retrieve GSK runtime error message

API introduced: V5R1

---

**gsk_secure_soc_init()—Negotiate a secure session**

**Syntax**

```c
#include <gskssl.h>

int gsk_secure_soc_init(gsk_handle my_session_handle);
```
The `gsk_secure_soc_init()` function is used to negotiate a secure session, using the attributes set for the SSL environment and the secure session. This API does the SSL handshake to the remote peer; upon successful completion, you have a secure session established.

**Parameters**

`my_session_handle` (Input)

The handle for this secure session that was obtained through the `gsk_secure_soc_open()` API call.

**Authorities**

Authorization of *R (allow access to the object) to the certificate store file and its associated files is required. Authorization of *X (allow use of the object) to each directory of the path name of the certificate store file and its associated files is required.

**Return Value**

`gsk_secure_soc_init()` returns an integer. Possible values are:

- **[GSK_OK]**
  - `gsk_secure_soc_init()` was successful.

- **[GSK_INVALID_HANDLE]**
  - The handle specified was not valid.

- **[GSK_KEYRING_OPEN_ERROR]**
  - Certificate store file could not be opened.

- **[GSK_ERROR_BAD_KEYFILE_LABEL]**
  - The specified certificate store label is not valid.

- **[GSK_ERROR_BAD_V3_CIPHER]**
  - An SSLV3 or TLSV1 cipher suite was specified that is not valid.

- **[GSK_ERROR_BAD_V2_CIPHER]**
  - An SSLV2 cipher suite was specified that is not valid.

- **[GSK_ERROR_NO_CIPHERS]**
  - No ciphers available or no ciphers were specified.

- **[GSK_ERROR_NO_CERTIFICATE]**
  - No certificate is available for SSL processing.

- **[GSK_ERROR_BAD_CERTIFICATE]**
  - The certificate is bad.

- **[SSL_ERROR_NOT_TRUSTED_ROOT]**
  - The certificate is not signed by a trusted certificate authority.

- **[GSK_KEYFILE_CERT_EXPIRED]**
  - The validity time period of the certificate has expired.

- **[GSK_ERROR_BAD_MESSAGE]**
  - A badly formatted message was received.

- **[GSK_ERROR_UNSUPPORTED]**
  - Operation is not supported by SSL.

- **[GSK_ERROR_BAD_PEER]**
  - The peer system is not recognized.
[GSK_ERROR_CLOSED]
   The SSL session ended.

[GSK_ERROR_CERT_VALIDATION]
   The certificate is not valid or was rejected by the GSK_CERT_VALIDATION_CALLBACK program.

[GSK_AS400_ERROR_NO_INITIALIZE]
   A successful gsk_environment_init() was not previously called with this handle.

[GSK_AS400_ERROR_TIMED_OUT]
   The value specified for the handshake timeout expired before the handshake completed.

[GSK_AS400_ERROR_NOT_TCP]
   The socket descriptor type is not SOCK_STREAM or the address family is not AF_INET or AF_INET6.

[GSK_AS400_ERROR_ALREADY_SECURE]
   The socket descriptor is already in use by another secure session.

[GSK_INSUFFICIENT_STORAGE]
   Unable to allocate storage for the requested operation.

[GSK_AS400_ERROR_INVALID_POINTER]
   The my_session_handle pointer is not valid.

[GSK_INTERNAL_ERROR]
   An unexpected error occurred during SSL processing.

[GSK_ERROR_IO]
   An error occurred in SSL processing, check errno value.

Error Conditions
When the gsk_secure_soc_init() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EIO]
   Input/output error.

[EINTR]
   Interrupted function call.

[EDEADLK]
   Resource deadlock avoided.

[ETERM]
   Operation terminated.

[EUNATCH]
   The protocol required to support the specified address family is not available at this time.

Any errno that can be returned by send() or recv() can be returned by this API. See Sockets APIs for a description of the errno values they return.

If an errno is returned that is not in this list, see "Errno Values for UNIX-Type Functions" on page 101 for a description of the errno.

Usage Notes
1. The gsk_secure_soc_init() function is valid only on sockets that have an address family of AF_INET or AF_INET6 and a socket type of SOCK_STREAM.

2. When doing the SSL handshake with a GSK_SESSION_TYPE value of GSK_SERVER_SESSION or GSK_SERVER_SESSION_WITH_CL_AUTH, the GSK_CONNECT_CIPHER_SPEC value will be the first
cipher found in the ordered `GSK_V3_CIPHER_SPECS` list that was also found in the cipher list provided by the client during the SSL handshake.

3. When doing the SSL handshake with a `GSK_SESSION_TYPE` value of `GSK_CLIENT_SESSION`, the cipher specification list will be sent to the server in the client hello in the order found in the `GSK_V3_CIPHER_SPECS` and/or `GSK_V2_CIPHER_SPECS` list, however the value from that list that is negotiated for `GSK_CONNECT_CIPHER_SPEC` is determined by the server policy.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Related Information

- "gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment on page 16—Set character information for a secure session or an SSL environment.
- "gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment on page 23—Set enumerated information for a secure session or an SSL environment.
- "gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment on page 28—Set numeric information for a secure session or an SSL environment.
- "gsk_secure_soc_close()—Close a secure session” on page 36—Close a secure session
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session”—Perform miscellaneous functions for a secure session
- "gsk_secure_soc_open()—Get a handle for a secure session” on page 43—Get a handle for a secure session
- "gsk_secure_soc_read()—Receive data on a secure session” on page 45—Receive data on a secure session
- "gsk_secure_soc_startInit()—Start asynchronous operation to negotiate a secure session” on page 48—Start asynchronous operation to negotiate a secure session
- "gsk_secure_soc_write()—Send data on a secure session” on page 60—Send data on a secure session
- "gsk_strerror()—Retrieve GSKit runtime error message” on page 62—Retrieve GSK runtime error message

API Introduced: V5R1

---

gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session

Syntax

```c
#include <gskssl.h>

int gsk_secure_soc_misc(gsk_handle my_session_handle, 
                        GSK_MISC_ID miscID);
```

Service Program Name: QSYS/QSOSSLSR
Default Public Authority: *USE
Threadsafe: Yes
The `gsk_secure_soc_misc()` function is used to perform miscellaneous functions for a secure session.

**Parameters**

**my_session_handle (Input)**

The handle for the secure session obtained from `gsk_secure_soc_open()` and after performing a `gsk_secure_soc_init()`.

**miscID (Input)**

One of the following operations:

- **GSK_RESET_CIPHER (100)** - Performs another SSL handshake for the SSL session identified by the `my_session_handle` parameter. If an SSL session’s cache entry is still valid and both end points of the SSL session allow using a cache entry, an abbreviated SSL handshake may be performed. If the SSL cache entry for this session has expired or if the SSL session’s cache entry has been reset with the GSK_RESET_SESSION function, or if one end point of the SSL session does not allow using the SSL session cache entry, then a full SSL handshake will be performed.

- **GSK_RESET_SESSION (101)** - Removes this set of SSL session attributes from the SSL session cache. Any new SSL session handshake requests to the peer end point will not use this set of attributes. In most cases, as result of this operation, a full SSL handshake will be performed for the next SSL handshake request between both end points.

**Authorities**

No authorization is required.

**Return Value**

`gsk_secure_soc_misc()` returns an integer. Possible values are:

- **[GSK_OK]**
  
  `gsk_secure_soc_misc()` was successful.

- **[GSK_INVALID_HANDLE]**
  
  The handle specified was not valid.

- **[GSK_INVALID_STATE]**
  
  A `gsk_secure_soc_init()` has not been issued with this handle.

- **[GSK_ERROR_NOT_SSLV3]**
  
  SSLV3 or TLSV1 is required for this function.

- **[GSK_MISC_INVALID_ID]**
  
  The value specified for `miscID` is not valid.

- **[GSK_AS400_ERROR_INVALID_POINTER]**
  
  The `my_session_handle` pointer is not valid.

- **[GSK_INTERNAL_ERROR]**
  
  An unexpected error occurred during SSL processing.

- **[GSK_ERROR_IO]**
  
  An error occurred in SSL processing; check the `errno` value.

- **[GSK_KEYRING_OPEN_ERROR]**
  
  Certificate store file could not be opened.

- **[GSK_ERROR_BAD_KEYFILE_LABEL]**
The specified certificate store label is not valid.

[GSK_ERROR_BAD_V3_CIPHER]
An SSLv3 or TLSv1 cipher suite was specified that is not valid.

[GSK_ERROR_BAD_V2_CIPHER]
An SSLv2 cipher suite was specified that is not valid.

[GSK_ERROR_NO_CIPHERS]
No ciphers available or no ciphers were specified.

[GSK_ERROR_NO_CERTIFICATE]
No certificate is available for SSL processing.

[GSK_ERROR_BAD_CERTIFICATE]
The certificate is bad.

[SSL_ERROR_NOT_TRUSTED_ROOT]
The certificate is not signed by a trusted certificate authority.

[GSK_KEYFILE_CERT_EXPIRED]
The validity time period of the certificate has expired.

[GSK_ERROR_BAD_MESSAGE]
A badly formatted message was received.

[GSK_ERROR_UNSUPPORTED]
Operation is not supported by SSL.

[GSK_ERROR_BAD_PEER]
The peer system is not recognized.

[GSK_ERROR_CLOSED]
The SSL session ended.

[GSK_AS400_ERROR_NO_INITIALIZE]
A successful gsk_environment_init() was not previously called with this handle.

[GSK_AS400_ERROR_TIMED_OUT]
The value specified for the handshake timeout expired before the handshake completed.

[GSK_AS400_ERROR_NOT_TCP]
The socket descriptor type is not SOCK_STREAM or the address family is not AF_INET or AF_INET6.

[GSK_AS400_ERROR_ALREADY_SECURE]
The socket descriptor is already in use by another secure session.

[GSK_INSUFFICIENT_STORAGE]
Unable to allocate storage for the requested operation.

Error Conditions
When the gsk_secure_soc_misc() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]
Interrupted function call.
Resource deadlock avoided.

Operation terminated.

Input/output error.

The protocol required to support the specified address family is not available at this time.

Usage Notes
1. An SSL session’s attributes that are negotiated as part of an SSL handshake may be cached by each end point involved in the SSL session and then reused as part of an abbreviated SSL handshake when allowed by both end points.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Related Information
- "gsk_secure_soc_close()—Close a secure session” on page 36—Close a secure session
- "gsk_secure_soc_init()—Negotiate a secure session” on page 37—Negotiate a secure session
- "gsk_secure_soc_open()—Get a handle for a secure session”—Get a handle for a secure session
- "gsk_strerror()—Retrieve GSKit runtime error message” on page 62—Retrieve GSKit runtime error message

API introduced: V5R1

Syntax

```c
#include <gskssl.h>

int gsk_secure_soc_open(gsk_handle my_env_handle,
                        gsk_handle *my_session_handle);
```

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

The `gsk_secure_soc_open()` function is used to get storage for a secure session, set default values for attributes, and return a handle that must be saved and used on secure session-related function calls.

Parameters

**my_env_handle (Input)**
- The handle for the SSL environment obtained from `gsk_environment_open()`.
my_session_handle (Output)
   Pointer to the secure session handle.

Authorities
No authorization is required.

Return Value
gsk_secure_soc_open() returns an integer. Possible values are:

[GSK_OK]
   gsk_secure_soc_open() was successful.

[GSK_INVALID_HANDLE]
   The environment handle specified was not valid.

[GSK_INSUFFICIENT_STORAGE]
   Not able to allocate storage for the requested operation.

[GSK_AS400_ERROR_INVALID_POINTER]
   The my_env_handle pointer is not valid.

[GSK_INTERNAL_ERROR]
   An internal error occurred during system processing.

[GSK_ERROR_IO]
   An error occurred in SSL processing, check the errno value.

Error Conditions
When the gsk_secure_soc_open() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EINTR]
   Interrupted function call.

[EDEADLK]
   Resource deadlock avoided.

[ETERM]
   Operation terminated.

If an errno is returned that is not in this list, see “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Usage Notes
1. After gsk_secure_soc_open() returns with a GSK_OK return value, attributes from the SSL environment will be used as the defaults for the subsequent gsk_secure_soc_init(). The defaults can be changed using the gsk_attribute_set_buffer(), gsk_attribute_set_enum(), or gsk_attribute_set_numeric_value() APIs after calling gsk_secure_soc_open() and before calling gsk_secure_soc_init().

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
</tbody>
</table>
Message ID  Error Message Text
CPF9872 E  Program or service program &1 in library &2 ended. Reason code &3.
CPFA081 E  Unable to set return value or error code.

Related Information

- “gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment” on page 16—Set character string information for a secure session or a SSL environment.
- “gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment” on page 23—Set enumerated information for a secure session or a SSL environment.
- “gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment” on page 28—Set numeric information for a secure session or a SSL environment.
- “gsk_environment_open()—Get a handle for an SSL environment” on page 34—Get a handle for a SSL environment.
- “gsk_secure_soc_close()—Close a secure session” on page 36—Close a secure session.
- “gsk_secure_soc_init()—Negotiate a secure session” on page 37—Negotiate a secure session.
- “gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40—Perform miscellaneous functions for a secure session.
- “gsk_strerror()—Retrieve GSK runtime error message” on page 62—Retrieve GSK runtime error message.

API introduced: V5R1

---

**gsk_secure_soc_read()—Receive data on a secure session**

Syntax

```c
#include <gskssl.h>

int gsk_secure_soc_read(gsk_handle my_session_handle,
                        char *readBuffer,
                        int readBufSize,
                        int *amtRead);
```

Service Program Name: QSYS/QSOSSL5R
Default Public Authority: *USE
Threadsafe: Yes

The **gsk_secure_soc_read()** function is used by a program to receive data from a secure session.

**Parameters**

- **my_session_handle (Input)**
  The handle, returned from **gsk_secure_soc_open()** and used on the **gsk_secure_soc_init()** API call that initialized the secure session over which data is to be read.

- **readBuffer (Output)**
  The pointer to the user-supplied buffer in which the data is to be stored.

- **readBufSize (Input)**
  The number of bytes to be read.

- **amtRead (Output)**
  The number of bytes that were read as a result of this API call.
Authorities
No authorization is required.

Return Value
`gsk_secure_soc_read()` returns an integer. Possible values are:

[GSK_OK]
    `gsk_secure_soc_read()` was successful.

[GSK_INVALID_HANDLE]
    The handle specified was not valid.

[GSK_INVALID_STATE]
    The handle is not in the correct state for this operation.

[GSK_INVALID_BUFFER_SIZE]
    The `readBufSize` is less than 1.

[GSK_WOULD_BLOCK]
    Operation would have caused the process to be suspended.

[GSK_ERROR_BAD_MESSAGE]
    SSL received a badly formatted message.

[GSK_ERROR_BAD_MAC]
    A bad message authentication code was received.

[GSK_OS400_ERROR_CLOSED]
    The secure session was closed by another thread before the read completed.

[GSK_OS400_ERROR_INVALID_POINTER]
    The `readBuffer` or `amtRead` pointer is not valid.

[GSK_OS400_ERROR_TIMED_OUT]
    The value specified for the receive timeout expired before the read completed.

[GSK_ERROR_SOCKET_CLOSED]
    A `close()` was done on the socket descriptor for this secure session.

[GSK_INTERNAL_ERROR]
    An unexpected error occurred during SSL processing.

[GSK_ERROR_IO]
    An error occurred in SSL processing; check the `errno` value.

Error Conditions
When the `gsk_secure_soc_read()` API fails with return code [GSK_ERROR_IO], `errno` can be set to:

[ECONNRESET]
    A connection with a remote socket was reset by that socket.

[EIO]
    Input/output error.

[ENOTCONN]
Requested operation requires a connection.

[EUNATCH]

The protocol required to support the specified address family is not available at this time.

Any *errno* that can be returned by `recv()` can be returned by this API. See [Sockets APIs](#) for a description of the *errno* values it can return.

If an *errno* is returned that is not in this list, see ["Errno Values for UNIX-Type Functions" on page 101](#) for a description of the *errno*.

**Usage Notes**

1. The maximum length of data typically returned will not exceed 16 KB. This is because SSL is a record level protocol and the largest record allowed is 32 KB minus the necessary SSL record headers.

2. It is strongly suggested that you do not mix the `gsk_secure_soc_read()` API with any of the sockets read functions. SSL and socket reads and writes can be mixed, but they must be performed in matched sets. If a client application writes 100 bytes of data using one or more of the socket send() calls, then the server application must read exactly 100 bytes of data using one or more of the socket `recv()` calls. This is also true for `gsk_secure_soc_read()` API.

3. Since SSL is a record-oriented protocol, SSL must receive an entire record before it can be decrypted and any data returned to the application. Thus, a select() may indicate that data is available to be read, but a subsequent `gsk_secure_soc_read()` may hang waiting for the remainder of the SSL record to be received when using blocking I/O.

4. A FIONREAD ioctl() cannot be used to determine the amount of data available for reading by using `gsk_secure_soc_read()`.

5. SSL will ignore the out-of-band (OOB) data indicator. OOB will not affect the SSL application. OOB will just be data to the SSL protocol.

6. For an SSL enabled socket, which must use a connection-oriented transport service (that is, TCP), a returned value of zero in the amtRead field indicates one of the following:
   - The partner program has issued a `close()` for the socket.
   - The partner program has issued a secure close for the secure session. For example, if the partner program was coded using the GSKit APIs, the partner issued `gsk_secure_soc_close()`.
   - The partner program has issued a `shutdown()` to disable writing to the socket.
   - The connection is broken and the error was returned on a previously issued socket function.
   - A `shutdown()` to disable reading was previously done on the socket.

7. When the secure session uses a blocking socket and `GSK_OS400_READ_TIMEOUT` was set, `GSK_OS400_ERROR_TIMED_OUT` will be the return value if no data arrives before the timeout expires.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

**Related Information**

- [“gsk_secure_soc_close()—Close a secure session” on page 36](#) — Close a secure session
- [“gsk_secure_soc_init()—Negotiate a secure session” on page 37](#) — Negotiate a secure session
- [“gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40](#) — Perform miscellaneous functions for a secure session
**gsk_secure_soc_startInit()**—Start asynchronous operation to negotiate a secure session

Syntax

```c
#include <gskssl.h>
#include <qsoasync.h>

int gsk_secure_soc_startInit(gsk_handle my_session_handle,
                          int IOCompletionPort,
                          Qso_OverlappedIO_t *communicationsArea)
```

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

The `gsk_secure_soc_startInit()` function is used to initiate an asynchronous negotiation of a secure session, using the attributes set for the SSL environment and the secure session. This API starts the SSL handshake to the remote peer and upon successful completion of `QsoWaitForIOCompletion()` a secure session is established.

**Parameters**

**my_session_handle** (Input)

The handle returned from `gsk_secure_soc_open()` that will be used to negotiate the secure session.

**int IOCompletionPort** (Input)

The I/O completion port that should be posted when the operation completes.

**Qso_OverlappedIO_t * communicationsArea** (Input/Output)

A pointer to a structure that contains the following information:

- **descriptorHandle** (Input) - The descriptor handle is application specific and is never used by the system. This field is intended to make it easier for the application to keep track of information regarding a given socket connection.
- **buffer** Not used.
- **bufferLength** Not used.
- **postFlag** Not used.
- **postFlagResult** Not used.
- **fillBuffer** Not used.
- **returnValue** (Output) - When the negotiate operation completes asynchronously, this field contains indication of success or failure.
- **errnoValue** (Output) - When the negotiate operation completes asynchronously and `returnValue` is `GSK_ERROR_IO`, this field will contain an `errno` further defining the failure.
operationCompleted
(Output) - If the operation is posted to the I/O completion port, this field is updated to indicate that the operation was a GSKSECURESOCSTARTINIT.

secureDataTransferSize
Not used.

bytesAvailable
Not used.

operationWaitTime
Not used.

postedDescriptor
Not used - Must be set to zero.

operationId
(Input) - An identifier to uniquely identify this operation or a group of operations. It can be set with the return value from QsoGenerateOperationId() or with an application-defined value. This value is preserved but ignored by all APIs except QsoCancelOperation() and QsoIsOperationPending().

reserved1
(Output) - Must be set to hexadecimal zeroes.

reserved2
(Input) - Must be set to hexadecimal zeroes.

Authorities
Authorization of *R (allow access to the object) to the certificate store file and its associated files is required. Authorization of *X (allow use of the object) to each directory of the path name of the certificate store file and its associated files is required.

Return Values

gsk_secure_soc_startInit() returns an integer. Possible values are:

- GSK_OS400ASYNCHRONOUS_SOC_INIT - The function has been started. When the function completes, the Qso_OverlappedIO_t communications structure will be updated with the results and the I/O completion port will be posted.
- If the function fails, possible values are:

  [GSK_INVALID_HANDLE]
  The handle specified was not valid.

  [GSK_OS400_ERROR_NO_INITIALIZE]
  A successful gsk_environment_init() was not previously called with this handle.

  [GSK_OS400_ERROR_NOT_TCP]
  The socket descriptor type is not SOCK_STREAM or the address family is not AF_INET or AF_INET6.

  [GSK_OS400_ERROR_ALREADY_SECURE]
  The socket descriptor is already in use by another secure session.

  [GSK_OS400_ERROR_INVALID_POINTER]
  The my_session_handle pointer is not valid.

  [GSK_INTERNAL_ERROR]
  An unexpected error occurred during SSL processing.

  [GSK_OS400_ERROR_INVALID_OVERLAPPEDIO_T]
  The Qso_OverLappedIO_t specified was not valid.

  [GSK_OS400_ERROR_INVALID_IOCOMPLETIONPORT]
  The I/O completion port specified was not valid.

  [GSK_OS400_ERROR_BAD_SOCKET_DESCRIPTOR]
  The socket descriptor specified within the gsk_handle was not valid.
[GSK_ERROR_IO]

An error occurred in SSL processing; check the errno value.

Error Conditions

When gsk_secure_soc_startInit() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EIO] Input/output error.

[EUNATCH]
The protocol required to support the specified address family is not available at this time.

If an errno is returned that is not in this list, see "Errno Values for UNIX-Type Functions" on page 101 for a description of the errno.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. The gsk_secure_soc_startInit() function is valid only on sockets that have an address family of AF_INET or AF_INET6 and a socket type of SOCK_STREAM.

2. The current implementation of the SSL Protocol does not allow gsk_secure_soc_startInit() to complete synchronously. Use gsk_secure_soc_startInit() if the synchronous behaviour is needed.

3. When doing the SSL handshake with a GSK_SESSION_TYPE value of GSK_SERVER_SESSION or GSK_SERVER_SESSION_WITH_CL_AUTH, the GSK_CONNECT_CIPHER_SPEC value will be the first cipher found in the ordered GSK_V3_CIPHER_SPECS( GSK_V2_CIPHER_SPECS if SSLV2 is only common protocol) list that was also found in the cipher list provided by the client during the SSL handshake.

4. When doing the SSL handshake with a GSK_SESSION_TYPE value of GSK_CLIENT_SESSION, the cipher specification list will be sent to the server in the client hello in the order found in the GSK_V3_CIPHER_SPECS and/or GSK_V2_CIPHER_SPECS list, however the value from that list that is negotiated for GSK_CONNECT_CIPHER_SPEC is determined by the server policy.

Related Information

- "gsk_secure_soc_close()—Close a secure session" on page 36—Close a secure session
- "gsk_secure_soc_init()—Negotiate a secure session" on page 37—Negotiate a secure session
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session" on page 40—Perform miscellaneous functions for a secure session
- "gsk_secure_soc_open()—Get a handle for a secure session" on page 43—Get a handle for a secure session
- "gsk_secure_soc_read()—Receive data on a secure session" on page 45—Receive data on a secure session
- "gsk_secure_soc_write()—Send data on a secure session" on page 60—Send data on a secure session
- "gsk_secure_soc_startRecv()—Start asynchronous receive operation on a secure session" on page 51—Start AsynchronousRecv Operation on a secure session
- "gsk_secure_soc_startSend()—Start asynchronous send operation on a secure session" on page 55—Start AsynchronousSend Operation on a secure session
- QsoCancelOperation()—Cancel an I/O Operation

50 IBM Systems - iSeries: UNIX-Type -- Secure Sockets APIs
gsk_secure_soc_startRecv()—Start asynchronous receive operation on a secure session

Syntax
#include <gskssl.h>
#include <qsoasync.h>

int gsk_secure_soc_startRecv (gsk_handle my_session_handle,
    int IOCompletionPort,
    Qso_OverlappedIO_t * communicationsArea)

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

The gsk_secure_soc_startRecv() function is used to initiate an asynchronous receive operation on a secure session. The supplied receive buffer cannot be reused by the calling application until the receive is complete or the I/O completion port specified on the gsk_secure_soc_startRecv() has been destroyed. This API supports sockets with an address family of AF_INET or AF_INET6 and type SOCK_STREAM only.

Parameters

my_session_handle (Input)
The handle, returned from gsk_secure_soc_open() and used on the gsk_secure_soc_init() API call that initialized the secure session over which data is to be read.

int IOCompletionPort (Input)
The I/O completion port that should be posted when the operation completes.

Qso_OverlappedIO_t * communicationsArea (Input/Output)
A pointer to a structure that contains the following information:

descriptorHandle (Input) - The descriptor handle is application specific and is never used by the system. This field is intended to make it easier for the application to keep track of information regarding a given socket connection.

buffer (Input) - A pointer to a buffer into which data should be read.

bufferLength (Input) - The length of the buffer into which data should be read. Also represents the amount of data requested.
**postFlag**

(Input) - The postFlag indicates if this operation should be posted to the I/O completion port even if it completes immediately.

- A 0 value indicates that if the operation is already complete upon return to the application, then do not post to the I/O completion port.
- A 1 value indicates that even if the operation completes immediately upon return to the application, the result should still be posted to the I/O completion port.

**postFlagResult**

(Output) - This field is valid if gsk_secure_soc_startRecv() returns with 1 and postFlag was set to 1. In this scenario, postFlagResult set to 1 denotes the operation completed and been posted to the I/O completion port specified. A value of 0 denotes the operation could not be completed immediately, but will be handled asynchronously.

**fillBuffer**

(Input) - The fillBuffer flag indicates when this operation should complete. If the fillBuffer flag is 0, then the operation will complete as soon as any data is available to be received. If the fillBuffer flag is non-zero, this operation will not complete until enough data has been received to fill the buffer, an end-of-file condition occurs on the socket, or an error occurs on a socket.

**returnValue**

(Output) - If gsk_secure_soc_startRecv() completes synchronously (function return value equals GSK_OK), then this field is set to GSK_OK and field secure data transfer size indicates number of bytes received.

**errnoValue**

(Output) - When the operation has completed asynchronously and returnValue is GSK_ERROR_IO, this field will contain an errno further defining the failure.

**operationCompleted**

(Output) - If the operation is posted to the I/O completion port, this field is updated to indicate that the operation was a gsksecuresocstartrecv.

**secureDataTransferSize**

(Output) - Number of bytes received when gsk_secure_soc_startRecv() completes synchronously (return value equals GSK_OK).

**bytesAvailable**

Not used.

**operationWaitTime**

(Input) - A timeval structure which specifies the maximum time allowed for this operation to complete asynchronously.

```c
struct timeval {
    long tv_sec; /* second */
    long tv_usec; /* microsecond */
};
```

If this timer expires, the operation will be posted to the I/O completion port with errnoValue set to EAGAIN.

If this field is set to zero, the operation's asynchronous completion will not be timed.

If socketDescriptor is closed before the operation completes or times out, the operation will be posted to the I/O completion port with errnoValue set to ECLOSED.

The minimum operationWaitTime is 1 second. The microseconds field (tv_usec) in the timeval is not used and must be set to zero.

**postedDescriptor**

Not used - Must be set to zero.

**operationId**

(Input) - An identifier to uniquely identify this operation or a group of operations. It can be set with the return value from QsoGenerateOperationId() or with an application-defined value.

This value is preserved but ignored by all APIs except QsoCancelOperation() and QsoIsOperationPending().

**reserved1**

(Output) - Must be set to hexadecimal zeroes.

**reserved2**

(Input) - Must be set to hexadecimal zeroes.

**Authorities**

No authorization is required.
Return Values

gsk_secure_soc_startRecv() returns an integer. Possible values are:

- **GSK_OK** - The function has completed synchronously. The Qso_OverlappedIO_t communications structure has been updated but nothing has nor will be posted to the I/O completion port for this operation. Inspect field secureDataTransferSize in the Qso_OverlappedIO_t communications structure to determine the number of bytes received.
- **GSK_AS400ASYNCHRONOUS_RECV** - The function has been started. When the function completes (or times out if operationWaitTime was specified), the Qso_OverlappedIO_t communications structure will be updated with the results and the I/O completion port will be posted.
- If the function fails, possible values are:

  - **[GSK_INVALID_HANDLE]**
    The handle specified was not valid.
  - **[GSK_INVALID_STATE]**
    The handle is not in the correct state for this operation.
  - **[GSK_INVALID_BUFFER_SIZE]**
    The bufferLength field located in the Qso_OverLappedIO_t communications area is less than 1.
  - **[GSK_ERROR_BAD_MESSAGE]**
    SSL received a badly formatted message.
  - **[GSK_ERROR_BAD_MAC]**
    A bad message authentication code was received.
  - **[GSK AS400_ERROR_INVALID_POINTER]**
    The buffer pointer located in Qso_OverLappedIO_t communications area is not valid.
  - **[GSK_ERROR_SOCKET_CLOSED]**
    A close() was done on the socket descriptor for this secure session.
  - **[GSK_INTERNAL_ERROR]**
    An unexpected error occurred during SSL processing.
  - **[GSK AS400_ERROR_INVALID_ overlappedIO_T]**
    The Qso_OverLappedIO_t specified was not valid.
  - **[GSK AS400_ERROR_INVALID_IOCOMPLETIONPORT]**
    The I/O completion port specified was not valid.
  - **[GSK AS400_ERROR_BAD_SOCKET_DESCRIPTOR]**
    The socket descriptor specified within the gsk_handle was not valid.
  - **[GSK_ERROR_IO]**
    An error occurred in SSL processing; check the **errno** value.

Error Conditions

When gsk_secure_soc_startRecv() API fails with return code [GSK_ERROR_IO], **errno** can be set to:

- **[ECONNRESET]**
  A connection with a remote socket was reset by that socket.
[EINVAL]

The field operationWaitTime.tv_sec was negative or operationWaitTime.tv_usec was not zero or postedDescriptor was not zero.

[EIO]

Input/output error.

[ENOTCONN]

Requested operation requires a connection.

[EUNATCH]

The protocol required to support the specified address family is not available at this time.

If an errno is returned that is not in this list, see “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. A buffer that is given to gsk_secure_soc_startRecv() must not be used by the application again until either it is returned by QsoWaitForIOCompletion() or is reclaimed by issuing a close() on the socket descriptor or issuing a QsoDestroyIOCompletionPort() on the I/O completion port. If a buffer is given to gsk_secure_soc_startRecv() to be filled, and it is later detected during gsk_secure_soc_startRecv() processing that the buffer has been freed, it may produce an unrecoverable condition on the socket for which the gsk_secure_soc_startRecv() was issued. If this occurs, an ECONNABORTED error value will be returned.

2. It is not recommended to intermix gsk_secure_soc_startRecv() and blocking I/O (ie, recv() or gsk_secure_soc_read()) on the same socket. If this condition occurs, then pending asynchronous recv I/O will be serviced first before the blocking I/O.

3. The maximum length of data typically returned will not exceed 16 KB. This is due to the fact that SSL is a record level protocol and the largest record allowed is 32 KB minus the necessary SSL record headers.

4. Socket option SO_RCVLOWAT is not supported by this API. Semantics similar to SO_RCVLOWAT can be obtained using the fillBuffer field in the Qso_OverLappedIO_t structure.

5. Socket option SO_RCVTIMEO is not supported by this API. Semantics similar to SO_RCVTIMEO can be obtained using the operationWaitTime field in the Qso_OverLappedIO_t structure.

6. It is strongly suggested that you do not mix the gsk_secure_soc_read() nor gsk_secure_soc_startRecv() APIs with any of the sockets read functions. However, SSL and socket reads and writes can be mixed, but they must be performed in matched sets. If a client application writes 100 bytes of data using one or more of the socket send() calls, then the server application must read exactly 100 bytes of data using one or more of the socket recv() calls. This is also true for gsk_secure_soc_read() and gsk_secure_soc_startRecv() APIs.

7. A FIONREAD ioctl() cannot be used to determine the amount of data available for reading by using gsk_secure_soc_startRecv().

8. SSL will ignore the out of band (OOB) data indicator. OOB will not affect the SSL application. OOB will only be data to the SSL protocol.
For an SSL enabled socket, which must use a connection-oriented transport service (that is, TCP), a returned value of zero in the secureDataTransferSize field indicates one of the following:

- The partner program has issued a close() for the socket.
- The partner program has issued a secure close for the secure session. For example, if the partner program was coded using the GSKit APIs, the partner issued gsk_secure_soc_close().
- The partner program has issued a shutdown() to disable writing to the socket.
- The connection is broken and the error was returned on a previously issued socket function.
- A shutdown() to disable reading was previously done on the socket.

Related Information

- "gsk_secure_soc_close()—Close a secure session" on page 36—Close a Secure Session
- "gsk_secure_soc_init()—Negotiate a secure session" on page 37—Negotiate a secure session
- "gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session" on page 40—Perform miscellaneous functions for a secure session
- "gsk_secure_soc_open()—Get a handle for a secure session” on page 43—Get a handle for a secure session
- "gsk_secure_soc_startInit()—Start asynchronous operation to negotiate a secure session” on page 48—Start Asynchronous Operation to negotiate a secure session
- "gsk_secure_soc_startSend()—Start asynchronous send operation on a secure session”—Start Asynchronous Send Operation on a secure session
- "gsk_secure_soc_write()—Send data on a secure session” on page 60—Send data on a secure session

Service Program Name: QSYS/QSOSSLSR
Default Public Authority: *USE
Threadsafe: Yes
The `gsk_secure_soc_startSend()` function is used to initiate an asynchronous send operation on a secure session. The supplied send buffer cannot be reused by the calling application until the send is complete or the I/O completion port specified on the `gsk_secure_soc_startSend()` has been destroyed. This API supports sockets with an address family of AF_INET or AF_INET6 and type SOCK_STREAM only.

**Parameters**

**my_session_handle** *(Input)*
- The handle, returned from `gsk_secure_soc_open()` and used on the `gsk_secure_soc_init()` API call that initialized the secure session over which data is to be written.

**int IOCompletionPort** *(Input)*
- The I/O completion port that should be posted when the operation completes.

**Qso_OverlappedIO_t * communicationsArea** *(Input/Output)*
- A pointer to a structure that contains the following information:

  - `descriptorHandle` *(Input)* - The descriptor handle is application-specific and is never used by the system. This field is intended to make it easier for the application to keep track of information regarding a given socket connection.
  - `buffer` *(Input)* - A pointer to a buffer of data that should be sent over the socket.
  - `bufferLength` *(Input)* - The length of the data to be sent.
  - `postFlag` *(Input)* - The postFlag indicates if this operation should be posted to the I/O completion port even if it completes immediately:
    - A value of 0 indicates that if the operation is already complete upon return to the application, then do not post to the I/O completion port.
    - A value of 1 indicates that even if the operation completes immediately upon return to the application, the result should still be posted to the I/O completion port.
  - `postFlagResult` *(Output)* - This field is valid if `gsk_secure_soc_startSend()` returns with 1 and `postFlag` was set to 1. In this scenario, `postFlagResult` set to 1 denotes the operation completed and been posted to the I/O completion port specified. A value of 0 denotes the operation could not be completed immediately, but will be handled asynchronously.
  - `fillBuffer` *(Input)* - Only used on `gsk_secure_soc_startRecv()` or `QsoStartRecv()`. Ignored on `gsk_secure_soc_startSend()`.
  - `returnValue` *(Output)* - If `gsk_secure_soc_startSend()` completes synchronously (return value equals GSK_OK), then this field is set to GSK_OK and field `secureDataTransferSize` indicates number of bytes sent.
  - `errnoValue` *(Output)* - When the operation has completed asynchronously and returnValue is GSK_ERROR_IO, this field will contain an errno further defining the failure.
  - `operationCompleted` *(Output)* - If the operation is posted to the I/O completion port, this field is updated to indicate that the operation was a GSKSECUREOCSTARTSEND.
  - `secureDataTransferSize` *(Output)* - Number of bytes sent when `gsk_secure_soc_startSend()` completes synchronously (function return value equals GSK_OK).
  - `bytesAvailable` Not used.
**operationWaitTime**

(Input) - A timeval structure which specifies the maximum time allowed for this operation to complete asynchronously.

```c
struct timeval {
    long tv_sec; /* second */
    long tv_usec; /* microseconds */
};
```

If this timer expires, the operation will be posted to the I/O completion port with `errnoValue` set to EAGAIN.

If this field is set to zero, the operation’s asynchronous completion will not be timed.

If socketDescriptor is closed before the operation completes or times out, the operation will be posted to the I/O completion port with `errnoValue` set to ECLOSED.

The minimum operationWaitTime is 1 second. The microseconds field (tv_usec) in the timeval is not used and must be set to zero.

**postedDescriptor**

Not used - Must be set to zero.

**operationId**

(Input) - An identifier to uniquely identify this operation or a group of operations. It can be set with the return value from `QsoGenerateOperationId()` or with an application-defined value.

This value is preserved but ignored by all APIs except `QsoCancelOperation()` and `QsoIsOperationPending()`.

**reserved1**

(Input) - Must be set to hex zeroes.

**reserved2**

(Input) - Must be set to hex zeroes.

**Authorities**

No authorization is required.

**Return Values**

gsk_secure_soc_startSend() returns an integer. Possible values are:

- **GSK_OK** - The function has completed synchronously. The Qso_OverlappedIO_t communications structure has been updated but nothing has nor will be posted to the I/O completion port for this operation. Inspect field secureDataTransferSize in the Qso_OverlappedIO_t communications structure to determine the number of bytes sent.

- **GSK AS400 ASYNCHRONOUS_SEND** - The function has been started. When the function completes (or times out if operationWaitTime was specified), the Qso_OverlappedIO_t communications structure will be updated with the results and the I/O completion port will be posted.

- If the function fails, possible values are:
  
  **[GSK_INVALID_HANDLE]**

  The handle specified was not valid.

  **[GSK_INVALID_STATE]**

  The handle is not in the correct state for this operation.

  **[GSK_INVALID_BUFFER_SIZE]**

  The bufferLength field located in the Qso_OverLappedIO_t communications area is less than 1.

  **[GSK_ERROR_SOCKET_CLOSED]**

  A close() was done on the socket descriptor for this secure session.
The buffer pointer located in Qso_OverlappedIO_t communications area is not valid.

An unexpected error occurred during SSL processing.

The Qso_OverlappedIO_t specified was not valid.

The I/O completion port specified was not valid.

The socket descriptor specified within the gsk_handle was not valid.

An error occurred in SSL processing; check the errno value.

Error Conditions
When gsk_secure_soc_startSend() API fails with return code [GSK_ERROR_IO], errno can be set to:

The field operationWaitTime.tv_sec was negative or operationWaitTime.tv_usec was not zero or postedDescriptor was not zero.

Input/output error.

Requested operation requires a connection.

The specified descriptor does not reference a socket.

Broken pipe.

The protocol required to support the specified address family is not available at this time.

If an errno is returned that is not in this list, see “Errno Values for UNIX-Type Functions” on page 101 for a description of the errno.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes
1. Since gsk_secure_soc_startSend() is asynchronous, care should be used to control how many of these functions are outstanding. When a TCP socket becomes flow control blocked such that the
gsk_secure_soc_startSend() is not able to pass the data to the TCP socket immediately, the return value will be GSK_ASYNCHRONOUS_SEND. Applications that send large amounts of data should have the postFlag set to 0. This allows the application to use a return value of GSK_ASYNCHRONOUS_SEND as an indication that the socket has become flow control blocked. The application should then wait for the outstanding operation to complete before issuing another gsk_secure_soc_startSend(). This will ensure that the application does not exhaust system buffer resources.

2. A buffer that is given to gsk_secure_soc_startSend() must not be used by the application again until either it is returned by QsoWaitForICompletion() or is reclaimed by issuing a close() on the socket descriptor or issuing a QsoDestroyIOCompletionPort() on the I/O completion port. If a buffer is given to gsk_secure_soc_startSend() to be sent, and it is later detected during gsk_secure_soc_startSend() processing that the buffer has been freed, it may produce an unrecoverable condition on the socket for which the gsk_secure_soc_startSend() was issued. If this occurs, an ECONNABORTED error value will be returned.

3. There is no maximum length of data that can be written.

4. It is not recommended to intermix gsk_secure_soc_startSend() and blocking I/O (ie, send() or gsk_secure_soc_send()) on the same socket. If one does, then pending asynchronous send I/O will be serviced before blocking I/O once data can be sent.

5. It is strongly suggested that you do not mix the gsk_secure_soc_write() nor gsk_secure_soc_startSend() APIs with any of the sockets write functions. However, SSL and socket reads and writes can be mixed, but they must be performed in matched sets. If a client application writes 100 bytes of data using one or more of the socket send() calls, then the server application must read exactly 100 bytes of data using one or more of the socket recv() calls. This is also true for gsk_secure_soc_write() and gsk_secure_soc_startSend() APIs.

6. Socket option SO_SNDTIMEO is not supported by this API. Semantics similar to SO_SNDTIMEO can be obtained using the operationWaitTime field in the Qso_OverLappedIO_t structure.

Related Information

- gsk_secure_soc_init()—Negotiate a secure session on page 37—Negotiate a secure session
- gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session on page 40—Perform miscellaneous functions for a secure session
- gsk_secure_soc_open()—Get a handle for a secure session on page 43—Get a handle for a secure session
- gsk_secure_soc_read()—Receive data on a secure session on page 45—Receive data on a secure session
- gsk_secure_soc_startInit()—Start asynchronous operation to negotiate a secure session on page 48—Start Asynchronous Operation to negotiate a secure session
- gsk_secure_soc_startRecv()—Start asynchronous receive operation on a secure session on page 51—Start Asynchronous Receive Operation on a secure session
- QsoCancelOperation()—Cancel an I/O Operation

- QsoPostIOCompletionPort()—Post Request on I/O Completion Port
- QsoCreateIOCompletionPort()—Create I/O Completion Port
- QsoDestroyIOCompletionPort()—Destroy I/O Completion Port
- QsoStartRecv()—Start Asynchronous Recv Operation
- QsoStartSend()—Start Asynchronous Send Operation
- QsoWaitForIOCompletion()—Wait for I/O Completion Operation
gsk_secure_soc_write()—Send data on a secure session

Syntax
#include <gskssl.h>

int gsk_secure_soc_write(gsk_handle my_session_handle,
                        char *writeBuffer,
                        int writeBufSize,
                        int *amtWritten);

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

The gsk_secure_soc_write() function is used by a program to write data on a secure session.

Parameters

my_session_handle (Input)
The handle, returned from gsk_secure_soc_open() and used on the gsk_secure_soc_init() API call that initialized the secure session over which data is to be written.

writeBuffer (Input)
The pointer to the user-supplied buffer from which the data is to be written.

writeBufSize (Input)
The number of bytes to be written.

amtWritten (Output)
The number of bytes written as a result of this API call.

Authorities
No authorization is required.

Return Value

gsk_secure_soc_write() returns an integer. Possible values are:

[GSK_OK]
gsk_secure_soc_write() was successful.

[GSK_INVALID_HANDLE]
The handle specified was not valid.

[GSK_INVALID_STATE]
The handle is not in the correct state for this operation.

[GSK_INVALID_BUFFER_SIZE]
The readBufSize is less than 1.

[GSK_WOULD_BLOCK]
Operation would have caused the process to be suspended.

[GSK_ERROR_SOCKET_CLOSED]
A close() was done on the socket descriptor for this secure session.

[GSK_AS400_ERROR_CLOSED]
The secure session was closed by another thread before the write completed.

[GSK_AS400_ERROR_INVALID_POINTER]
The writeBuffer or amtWritten pointer is not valid.

[GSK_INTERNAL_ERROR]
An unexpected error occurred during SSL processing.

[GSK_ERROR_IO]
An error occurred in SSL processing; check the errno value.

Error Conditions
When the gsk_secure_soc_write() API fails with return code [GSK_ERROR_IO], errno can be set to:

[EIO]
Input/output error.

[ENOTCONN]
Requested operation requires a connection.

[ENOTSOCK]
The specified descriptor does not reference a socket.

[EPIPE]
Broken pipe.

[EUNATCH]
The protocol required to support the specified address family is not available at this time.

Any errno that can be returned by send() can be returned by this API. See Sockets APIs for a description of the errno values it can return.

Usage Notes
1. There is no maximum length of the data that can be written.
2. It is strongly suggested that you do not mix the gsk_secure_soc_write() API with any of the sockets write functions. SSL and socket reads and writes can be mixed, but they must be performed in matched sets. If a client application writes 100 bytes of data using one or more of the socket send() calls, then the server application must read exactly 100 bytes of data using one or more of the socket recv() calls. This is also true for gsk_secure_soc_write() API.
3. The amtWritten value is set to zero when return value is not GSK_OK.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Related Information

- "gsk_secure_soc_init()—Negotiate a secure session" on page 37—Negotiate a secure session
gsk_strerror()—Retrieve GSK runtime error message

Syntax
#include <gskssl.h>

const char *gsk_strerror(int gsk_return_value);

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

The gsk_strerror() function is used to retrieve an error message and associated text string that describes a return value that was returned from calling a GSKit API.

Parameters

gsk_return_value (Input)
The return value received from a GSKit API.

Authorities
No authorization is required.

Return Value

gsk_strerror() returns a pointer to the return value text.

Usage Notes

1. gsk_strerror() returns a pointer to the string. The null-terminated string is stored in the CCSID of the job.
2. If a gsk_return_value is specified for which there is no corresponding description, an Unknown Error string is returned.

Related Information

- "gsk_attribute_get_buffer()—Get character information about a secure session or an SSL environment” on page 3—Get character information about a secure session or an SSL environment.
- "gsk_attribute_get_cert_info()—Get information about a local or partner certificate” on page 6—Get information about a local or partner certificate.
- "gsk_attribute_get_enum()—Get enumerated information about a secure session or an SSL environment>” on page 10—Get enumerated information about a secure session or an SSL environment.
gsk_attribute_get_numeric_value()—Get numeric information about a secure session or an SSL environment on page 14—Get numeric information about a secure session or an SSL environment.

“gsk_attribute_set_buffer()—Set character information for a secure session or an SSL environment” on page 16—Set character information for a secure session or an SSL environment.

“gsk_attribute_set_enum()—Set enumerated information for a secure session or an SSL environment” on page 23—Set enumerated information for a secure session or an SSL environment.

“gsk_attribute_set_numeric_value()—Set numeric information for a secure session or an SSL environment” on page 28—Set numeric information for a secure session or an SSL environment.

“gsk_environment_close()—Close an SSL environment” on page 30—Close the SSL environment

“gsk_environment_init()—Initialize an SSL environment” on page 32—Initialize a SSL environment

“gsk_environment_open()—Get a handle for an SSL environment” on page 34—Get a handle for an SSL environment

“gsk_secure_soc_close()—Close a secure session” on page 36—Close a secure session

“gsk_secure_soc_init()—Negotiate a secure session” on page 37—Negotiate a secure session

“gsk_secure_soc_misc()—Perform miscellaneous functions for a secure session” on page 40—Perform miscellaneous functions for a secure session

“gsk_secure_soc_open()—Get a handle for a secure session” on page 43—Get a handle for a secure session

“gsk_secure_soc_read()—Receive data on a secure session” on page 45—Receive data on a secure session

“gsk_secure_soc_write()—Send data on a secure session” on page 60—Send data on a secure session

Example

See Code disclaimer information for information pertaining to code examples.

The following example shows how gsk_strerror() is used:

```c
#include <stdio.h>
#include <sys/types.h>
#include <gskssl.h>

void main()
{
    int rc = GSK_OK;
    gsk_handle env_handle = NULL;

    rc = gsk_environment_open(&env_handle);
    if (rc != GSK_OK)
    {
        printf("gsk_environment_open() failed with rc = %d %s\n", rc,gsk_strerror(rc));
        break;
    }

    ...
}
```

API introduced: V5R1
i5/OS Secure Sockets Layer (SSL_) APIs

i5/OS(I)M SSL_ APIs, when used in addition to the existing i5/OS Sockets APIs, provide the functions required for applications to establish secure communications. An application using SSL for secure communications is basically a client/server application written using sockets.

The SSL_ APIs are:

- "QlgSSL_Init()—Initialize the Current Job for SSL (using NLS-enabled path name)" (Initialize the current job for SSL (using NLS-enabled path name)) is used to establish the SSL security information to be used for all SSL sessions for the current job.
- "SSL_Create()—Enable SSL Support for the Specified Socket Descriptor” on page 68 (Enable SSL support for the specified socket descriptor) is used by a program to enable SSL support for the specified socket descriptor.
- "SSL_Destroy()—End SSL Support for the Specified SSL Session” on page 71 (End SSL support for the specified SSL session) is used by a program to end SSL support for the specified SSL session.
- "SSL_Init()—Initialize the Current Job for SSL” on page 79 (Initialize the current job for SSL) is used to establish the SSL security information to be used for all SSL sessions for the current job.
- "SSL_Init_Application()—Initialize the Current Job for SSL Processing Based on the Application Identifier” on page 83 (Establish the SSL security information) is used to establish the SSL security information to be used for all SSL sessions for the current job based on the specified application identifier.
- "SSL_Perror()—Print SSL Error Message” on page 87 (Print SSL error message) prints an error message to stderr.
- "SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor” on page 89 (Receive data from an SSL-enabled socket descriptor) is used by a program to receive data from an SSL-enabled socket descriptor.
- "SSL_Sstrerror()—Retrieve SSL Runtime Error Message” on page 93 (Retrieve SSL runtime error message) is used to retrieve an error message and associated text string which describes an SSL return value.
- "SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor” on page 95 (Write data to an SSL-enabled socket descriptor) is used by a program to write data to an SSL-enabled socket descriptor.

Note: These functions use header (include) files from the library QSYSINC, which is optionally installable. Make sure QSYSINC is installed on your system before using any of the functions. See "Header Files for UNIX-Type Functions” on page 99 for the file and member name of each header file.

See the following examples for more information:

- Example: Establish secure server with SSL APIs
- Example: Establish secure client with SSL APIs

QlgSSL_Init()—Initialize the Current Job for SSL (using NLS-enabled path name)

Syntax
#include <qss1.h>

int QlgSSL_Init(QlgSSLInit* init)
The QlgSSL_Init() function is used to establish the SSL security information to be used for all SSL sessions for the current job. The QlgSSL_Init() API establishes a certificate and private key for use by the handshake protocol processing when acting as a server. The QlgSSL_Init() API establishes a certificate for use by the handshake protocol processing when acting as a client that is connected to a server performing client authentication.

Parameters

QlgSSLInit * init (input)

The pointer to a QlgSSLInit structure. QlgSSLInit is a typedef for a buffer of type struct QlgSSLInitStr. In <qsossl.h>, struct QlgSSLInitStr is defined as the following:

```c
struct QlgSSLInitStr {
    /* QlgSSLInitStr */
    Qlg_Path_Name* keyringFileName; /* Key ring file name */
    char* keyringPassword; /* Key ring file password */
    unsigned short int* cipherSuiteList; /* List of cipher suites */
    unsigned int cipherSuiteListLen; /* number of entries in the cipher suites list */
};
```

The fields within the QlgSSLInit structure as pointed to by init are defined as follows:

**Qlg_Path_Name_T* keyringFileName (input)**

A pointer to a structure defining the path to the key ring file. This structure defines the coded character set identifier (CCSID) and the path to the key ring file to be used for this job’s SSL processing. The path must be a fully qualified integrated file system file name.

**char* keyringPassword (input)**

A pointer to the password for the key ring file named in the keyringFileName field.

If this parameter’s value is equal to NULL, then the QlgSSL_Init() support will attempt to extract a password from a key-ring password file.

This parameter is assumed to be represented in the same CCSID (coded character set identifier) as the keyringFileName.

**unsigned short int* cipherSuiteList (input)**

A pointer to the cipher specification list to be used during the SSL handshake protocol for this job. This list is a string of concatenated cipher specification values. A cipher specification value is an unsigned short integer. Any value provided will override any values provided by a previous QlgSSL_Init() API or the system default cipher specification list if the previous QlgSSL_Init() API did not provide a cipher specification list. A value of NULL for this parameter indicates one of the following:

- Use the cipher specification list provided by a previous QlgSSL_Init() API
- Use the system default cipher specification list if a previous QlgSSL_Init() API was not done

The caller specifies the preferred order of the cipher specifications. The cipher specification values are defined in <qsossl.h> as the following:

- TLS_RSA_WITH_NULL_MD5 0x0001
- TLS_RSA_WITH_NULL_SHA 0x0002
- TLS_RSA_EXPORT_WITH_RC4_40_MD5 0x0003
- TLS_RSA_WITH_RC4_128_MD5 0x0004
- TLS_RSA_WITH_RC4_128_SHA 0x0005
Notes:
1. The SSL_RSA_EXPORT_WITH_DES40_CBC_SHA cipher is not supported by i5/OS.
2. The default cipher suite list in preference order is as follows:

```
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_DES_CBC_MD5
TLS_RSA_WITH_3DES_EDE_CBC_MD5
TLS_RSA_WITH_RC2_CBC_128_MD5
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
```

`unsigned int cipherSuiteListLen` (input)

The number of cipher suite entries specified in the list pointed to by the `cipherSuiteList` parameter.

Authorities

Authorization of *R (allow access to the object) to the key ring file and its associated files is required.

Return Value

The QlgSSL_Init() API returns an integer. Possible values are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful return</td>
</tr>
<tr>
<td>-</td>
<td>[SSL_ERROR_BAD_CIPHER_SUITE] A cipher suite that is not valid was specified.</td>
</tr>
<tr>
<td>-</td>
<td>[SSL_ERROR_IO] An error occurred in SSL processing; check the <code>errno</code> value.</td>
</tr>
<tr>
<td>-</td>
<td>[SSL_ERROR_KEYPASSWORD_EXPIRED] The specified key ring password has expired.</td>
</tr>
<tr>
<td>-</td>
<td>[SSL_ERROR_NO_KEYRING] No key ring file was specified.</td>
</tr>
<tr>
<td>-</td>
<td>[SSL_ERROR_SSL_NOT_AVAILABLE] SSL is not available for use.</td>
</tr>
<tr>
<td>-</td>
<td>[SSL_ERROR_UNKNOWN] An unknown or unexpected error occurred during SSL processing.</td>
</tr>
</tbody>
</table>

Error Conditions

When the QlgSSL_Init() API fails with return code [SSL_ERROR_IO], `errno` can be set to:
Parameter not valid.

This error code indicates that the Qlg_Path_Name_T structure was not valid:

- The path type was less than 0 or greater than 3.
- A reserved field was not initialized to zeros.

Conversion error.

This error code indicates one of the following:

- The CCSID specified in the keyringFileName cannot be converted to the current default CCSID for integrated file system path names.
- There was an incomplete character or shift state sequence at the end of the keyringFileName path or keyringPassword.

Permission denied.

This error code indicates one of the following:

- The keyringFileName field contains a file name to which the user is not authorized.
- The keyringPassword value is not valid for the specified keyringFileName.

Descriptor not valid.

This error code indicates one of the following:

- The keyringFileName value does not specify a valid key ring file name.

Bad address.

The system detected an address that was not valid while attempting to access the init parameter or one of the address fields in the init parameter.

The protocol required to support the specified address family is not available at this time.

Unknown system state.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. A successful SSL_Init(), QlgSSL_Init (using NLS-enabled path name), or SSL_Init_Application() API must be used to enable a job for SSL processing before attempting to use any other SSL API.
2. If multiple SSL_Init_Application(), QlgSSL_Init (using NLS-enabled path name), or SSL_Init() APIs are performed in a job, then only the values associated with the last SSL_Init_Application(), QlgSSL_Init (using NLS-enabled path name), or SSL_Init() performed are used.
3. If the keyringPassword parameter pointer value is equal to NULL, then QlgSSL_Init will attempt to extract the password value from the key-ring password file associated with the keyringFileName parameter’s value. The existence of the associated key-ring password file is based on a configuration selection made during the creation of the key ring file.
Related Information

- “SSL_Create()—Enable SSL Support for the Specified Socket Descriptor”—Enable SSL Support for the Specified Socket Descriptor
- “SSL_Destroy()—End SSL Support for the Specified SSL Session” on page 71—End SSL Support for the Specified SSL Session
- “SSL_Handshake()—Initiate the SSL Handshake Protocol” on page 73—Initiate the SSL Handshake Protocol
- “SSL_Init()—Initialize the Current Job for SSL” on page 79—Initialize the Current Job for SSL
- “SSL_Init_Application()—Initialize the Current Job for SSL Processing Based on the Application Identifier” on page 83—Initialize the Current Job for SSL Processing Based on the Application Identifier
- “SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor” on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- “SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor” on page 95—Write Data to an SSL-Enabled Socket Descriptor

API introduced: V5R1

SSL_Create()—Enable SSL Support for the Specified Socket Descriptor

Syntax

```
#include <qsossl.h>

SSLHandle* SSL_Create(int socket_descriptor,
                        int flags)
```

Service Program Name: QSOSSLSR
Default Public Authority: *USE
Threadsafe: Yes

The SSL_Create() function is used by a program to enable SSL support for the specified socket descriptor.

Parameters

- **int socket_descriptor** (input)
  The descriptor of the socket to be used for the SSL session. The socket descriptor must have been created (using the `socket()` API) with a type of SOCK_STREAM and an address family of AF_INET or AF_INET6.

- **int flags** (input)
  A flag value that controls the use of SSL for the session. The flags value is either zero, or is obtained by the ORing of the following constant:

  - `SSL_ENCRYPT (1<<0)` Encrypt the connection.
  - `SSL_DONT_ENCRYPT (0)` Do not encrypt the connection.

Authorities

No authorization is required.
Return Value

The SSL_Create() API returns a pointer to an SSLHandle. A value of NULL is returned when SSL_Create() fails. An SSLHandle is a typedef for a buffer of type struct SSLHandleStr. In <qsossl.h>, struct SSLHandleStr is defined as the following:

```c
struct SSLHandleStr {  /* SSLHandleStr */  int fd;  /* Socket descriptor */  int createFlags;  /* SSL_Creat flags value */  unsigned protocol;  /* SSL protocol version */  unsigned timeout;  /* Timeout value in seconds */  unsigned char cipherKind[3];  /* Current 2.0 cipher suite*/  unsigned short int cipherSuite;  /* Current 3.0 cipher suite */  unsigned short int* cipherSuiteList;  /* List of cipher suites */  unsigned int cipherSuiteListLen;  /* Number of entries in the cipher suites list */  unsigned char* peerCert;  /* Peer certificate */  unsigned peerCertLen;  /* Peer certificate length */  int peerCertValidateRc;  /* Return code from validation of certificate */  int (*exitPgm)(struct SSLHandleStr* sslh);  /* Authentication exit program called when a certificate is received during SSL handshake */
};
```

Note: A full explanation of each of the members of the above structure are defined in the SSL_Handshake() API description.

The SSLHandle structure returned will be initialized to hexadecimal zeros with the exception of the fd field, which will be initialized to the socket descriptor input parameter and the createFlags field, which will be initialized to the flags input parameter.

Error Conditions

When the SSL_Create() API fails, errno can be set to:

- **[EALREADY]**
  
  Operation already in progress.

- **[EBADF]**
  
  Descriptor not valid.

- **[EFAULT]**
  
  Bad address.

- **[EINVAL]**
  
  Parameter not valid.

  This error code indicates one of the following:
  - The socket descriptor type is not SOCK_STREAM or address family is not AF_INET or AF_INET6.
  - One of the parameters passed is not valid or is NULL.

- **[EIO]**
  
  Input/output error.

- **[ENOBUFS]**
  
  There is not enough buffer space for the requested operation.
The specified descriptor does not reference a socket.

Broken pipe.

The protocol required to support the specified address family is not available at this time.

Unknown system state.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

**Usage Notes**

1. The `SSL_Create()` function is only valid on sockets that have an address family of `AF_INET` or `AF_INET6` and a socket type of `SOCK_STREAM`. If the descriptor pointed to by the `socket_descriptor` parameter does not have the correct address family and socket type, `[SSL_ERROR_IO]` is returned and the `errno` value is set to EINVAL.

2. If the `flags` parameter specifies a value that does not include the SSL_ENCRYPT flag, then the SSL protocol will not be used for the connection. Not using the SSL protocol has the following effects:

   - The `SSL_Handshake()` API will simply return successful without performing any function.
   - The `SSL_Read()` API will simply call the sockets `read()` API with the same set of input parameters.
   - The `SSL_Write()` API will simply call the sockets `write()` API with the same set of input parameters.

3. Any use of `givedescriptor()` and `takedescriptor()` APIs must be performed prior to issuing an `SSL_Create()`.

**Related Information**

- “SSL_Destroy()—End SSL Support for the Specified SSL Session” on page 71—End SSL Support for the Specified SSL Session
- “SSL_Handshake()—Initiate the SSL Handshake Protocol” on page 73—Initiate the SSL Handshake Protocol
- “SSL_Init()—Initialize the Current Job for SSL” on page 79—Initialize the Current Job for SSL
- “SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor” on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- “SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor” on page 95—Write Data to an SSL-Enabled Socket Descriptor

API introduced: V4R3
SSL_Destroy()—End SSL Support for the Specified SSL Session

Syntax
#include <qsossl.h>

int SSL_Destroy(SSLHandle* handle)

Service Program Name: QSOSSLSR
Default Public Authority: *USE
Threadsafe: Yes

The SSL_Destroy() function is used by a program to end SSL support for the specified SSL session. The SSL session to be ended is identified by the handle parameter.

Parameters
SSLHandle* handle (input)
The pointer to an SSLHandle for an active SSL session, which is being ended. An SSLHandle is a typedef for a buffer of type struct SSLHandleStr. In <qsossl.h>, struct SSLHandleStr is defined as the following:

```c
struct SSLHandleStr { /* SSLHandleStr */
    int fd; /* Socket descriptor */
    int createFlags; /* SSL_Create flags value */
    unsigned protocol; /* SSL protocol version */
    unsigned timeout; /* Timeout value in seconds */
    unsigned char cipherKind[3]; /* Current 2.0 cipher suite*/
    unsigned short int cipherSuite; /* Current 3.0 cipher suite */
    unsigned short int* cipherSuiteList; /* List of cipher suites */
    unsigned int cipherSuiteListLen; /* Number of entries in
    the cipher suites list */
    unsigned char* peerCert; /* Peer certificate */
    unsigned peerCertLen; /* Peer certificate length */
    int peerCertValidateRc; /* Return code from
    validation of certificate */
    int (*exitPgm)(struct SSLHandleStr* sslh); /* Authentication exit
    program called when a certificate is received during SSL handshake */
}
```

Authorities
No authorization is required.

Return Value
The SSL_Destroy() API returns an integer. Possible values are:

[0] Successful return

[SSL_ERROR_IO] An error occurred in SSL processing; check the errno value.

Error Conditions
When the SSL_Destroy() API fails with return code [SSL_ERROR_IO], errno can be set to:

[EBADF]
    Descriptor not valid.

[EFAULT]
Bad address.
The system detected an address that was not valid while attempting to access the handle parameter or a field within the structure pointed to by the handle parameter.

[EIO]
Input/output error.

[EINVAL]
Parameter not valid. This error code indicates one of the following:
• The socket_descriptor type is not SOCK_STREAM or address family is not AF_INET or AF_INET6.
• One of the parameters passed is not valid or is NULL.

[ENOTCONN]
Requested operation requires a connection.
This error code indicates that the socket_descriptor has not had SSL support enabled. This usually means that an SSL_Create() has not been completed for this socket_descriptor.

[ENOTSOCK]
The specified descriptor does not reference a socket.

[EPIPE]
Broken pipe.

[ETIMEDOUT]
A remote host did not respond within the timeout period.
This error code indicates that the SSL_Destroy() was unable to successfully complete the removal of SSL support on this socket_descriptor.

[EUNATCH]
The protocol required to support the specified address family is not available at this time.

[EUNKNOWN]
Unknown system state.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes
1. All storage referenced from any field within the structure pointed to by the handle parameter and the storage pointed to by the handle parameter itself will be freed upon a successful return.
2. Unpredictable results will occur if you attempt to use an SSL_Destroy() while sending or receiving data on the peer system.
3. If an SSL_Destroy() is not done, then the storage referenced by the handle parameter will not be freed until the job ends.
Note: A job end might cause a Licensed Internal Code log entry or error log entry if the *handle* parameter storage is not freed before the job ended.

4. If an *SSL_Destroy()* is not done, the storage referenced by the *handle* parameter will not be freed. This will result in a memory leak. A **memory leak** is the loss of a piece of system memory because it is not allocated to any process on the system.

**Related Information**

- "SSL_Create()—Enable SSL Support for the Specified Socket Descriptor" on page 68—Enable SSL Support for the Specified Socket Descriptor
- "SSL_Handshake()—Initiate the SSL Handshake Protocol"—Initiate the SSL Handshake Protocol
- "SSL_Init()—Initialize the Current Job for SSL" on page 79—Initialize the Current Job for SSL
- "SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor” on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- "SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor” on page 95—Write Data to an SSL-Enabled Socket Descriptor

API introduced: V4R3

---

**SSL_Handshake()—Initiate the SSL Handshake Protocol**

**Syntax**

```c
#include <qsossl.h>

int SSL_Handshake(SSLHandle* handle,
                  int how)
```

Service Program Name: QSOSSLSR

Default Public Authority: *USE

Threading: Yes

The **SSL_Handshake()** function is used by a program to initiate the SSL handshake protocol. Both the client and the server program must call the **SSL_Handshake** verb in order to initiate the handshake processing.

**Parameters**

*SSLHandle* `handle` (input/output)

The pointer to an **SSLHandle** for an SSL session. An **SSLHandle** is a typedef for a buffer of type struct **SSLHandleStr**. In `<qsossl.h>`, struct **SSLHandleStr** is defined as the following:

```c
struct SSLHandleStr {
    int fd;               /* SSLHandleStr */
    int createFlags;     /* Socket descriptor */
    unsigned protocol;   /* SSL_Create flags value */
    unsigned timeout;    /* SSL protocol version */
    unsigned char cipherKind[3]; /* Timeout value in seconds */
    unsigned short int cipherSuite; /* Current 2.0 cipher suite */
    unsigned short int* cipherSuiteList; /* Current 3.0 cipher suite */
    unsigned int cipherSuiteListLen; /* List of cipher suites */
    unsigned int cipherSuiteListLen; /* Number of entries in */
                                /* the cipher suites list */
    unsigned char* peerCert;  /* Peer certificate */
};
```
unsigned peerCertLen;  /* Peer certificate length */
int peerCertValidateRc;  /* Return code from validation of certificate */
int (*exitPgm)(struct SSLHandleStr* sslh);
    /* Authentication exit program called when a certificate is received during SSL handshake */
};

The fields within the SSLHandle structure as pointed to by handle are defined as follows:

int fd (input)  The socket descriptor of the connection for which the SSL handshake protocol is to be performed. This field was initialized by a prior SSL_Create() API.
int createFlags (input)  Whether or not the SSL protocol is to be used. If the field specifies a value that does not include the SSL_ENCRYPT flag, then this function will return success without performing the SSL handshake protocol. This field was initialized by a prior SSL_Create() API.
unsigned int protocol (input/output)  The type of SSL handshake protocol to be performed. The protocol(s) that are acceptable as the handshake protocol for this job. The following values may be specified for protocol and are defined in <qsossl.h>.

SSL_VERSION_CURRENT  0 (TLS with SSL Version 3.0 and SSL Version 2.0 compatibility)
SSL_VERSION_2  2 (SSL Version 2.0 only)
SSL_VERSION_3  3 (SSL Version 3.0 only)
TLS_VERSION_1  4 (TLS Version 1 only)
TLSV1_SSLV3  5 (TLS Version 1 with SSL Version 3.0 compatibility)

Upon return, this field will be set to reflect the protocol version actually negotiated. If the createFlags field specifies a value that does not include the SSL_ENCRYPT flag, then this field will be unchanged from its input value.

unsigned timeout (input)  The approximate number of seconds to wait for the SSL handshake protocol to complete. A value of 0 indicates to wait forever for the handshake to complete.

unsigned char cipherKind[3] (output)  The cipher kind (which is the SSL Version 2.0 cipher suite) negotiated by the handshake.

unsigned short int cipherSuite (output)  The cipher suite type negotiated by the handshake.

unsigned short int* cipherSuiteList (input)  A pointer to a cipher specification list that is to be used during the handshake negotiation for this SSL session. This list is a string of concatenated cipher specification values. Each cipher specification is an unsigned short integer value. Any value provided will override, for this SSL session, the default cipher specification list provided by a previous SSL_Init() API or SSL_Init_Application() API. The valid cipher suites allowed are defined in <qsossl.h>. A value of NULL indicates one of the following:

- Use the cipher specification list provided by a previous SSL_Init() API or SSL_Init_Application() API
- Use the system default cipher specification list if the previous SSL_Init() API or SSL_Init_Application() API did not provide a cipher specification list

unsigned int cipherSuiteListLen (input)  The number of cipher suite entries specified in the list pointed to by the cipherSuiteList field.

unsigned char* peerCert (output)  The pointer to the certificate received from the peer system. For a client, this is a pointer to the server’s certificate. For a server with client authentication enabled, this is a pointer to the client’s certificate. For a server without client authentication this pointer value remains unchanged.
unsigned peerCertLen (output)  The length of the certificate pointed to by the peerCert field.
int (*exitPgm)(SSLHandle* sslh) (input)  A pointer to a user supplied function that is called whenever a certificate is received during handshake processing. The exitPgm will be passed the pointer to the handle, which could include the peer’s certificate. The exitPgm returns a nonzero value if the peer’s certificate is accepted. The return of a zero value by the exitPgm will cause the handshake processing to fail. Users of this function do not need to provide an exit program. The pointer should be a NULL value if there is not a user-supplied exit program. The exit program should be written in a threadsafe manner.

int how (input)  The type of SSL handshake to be performed. The following values may be specified for handshake type and are defined in <qsoss1.h>.

SSL_HANDSHAKE_AS_CLIENT (0)  Perform the handshake as a client.
SSL_HANDSHAKE_AS_SERVER (1)  Perform the handshake as a server.
SSL_HANDSHAKE_AS_SERVER_WITH_CLIENT_AUTH (2)  Perform the handshake as a server with client authentication.
SSL_HANDSHAKE_AS_SERVER_WITH_OPTIONAL_CLIENT_AUTH (3)  Perform the handshake as a server with optional client authentication.

Authorities
Authorization of *R (allow access to the object) to the key ring file and its associated files is required.

Return Value
The SSL_Handshake() API returns an integer. Possible values are:

[0]  Successful return

[SSL_ERROR_BAD_CERTIFICATE]  The certificate is bad.

[SSL_ERROR_BAD_CERT_SIG]  The certificate’s signature is not valid.

[SSL_ERROR_BAD_CIPHER_SUITE]  A cipher suite that is not valid was specified.

[SSL_ERROR_BAD_MAC]  A bad message authentication code was received.

[SSL_ERROR_BAD_MALLOC]  Unable to allocate storage required for SSL processing.

[SSL_ERROR_BAD_MESSAGE]  SSL received a badly formatted message.

[SSL_ERROR_BAD_PEER]  The peer system is not recognized.
[SSL_ERROR_BAD_STATE]
SSL detected a bad state in the SSL session.

[SSL_ERROR_CERTIFICATE_REJECTED ]
The certificate is not valid or was rejected by the exit program.

[SSL_ERROR_CERT_EXPIRED]
The validity time period of the certificate is expired.

[SSL_ERROR_CLOSED]
The SSL session ended.

[SSL_ERROR_IO]
An error occurred in SSL processing; check the errno value.

[SSL_ERROR_NO_CERTIFICATE]
No certificate is available for SSL processing.

[SSL_ERROR_NO_CIPHERS]
No ciphers available or specified.

[SSL_ERROR_NO_INIT]
SSL_Init() was not previously called for this job.

[SSL_ERROR_NOT_TRUSTED_ROOT]
The certificate is not signed by a trusted certificate authority.

[SSL_ERROR_PERMISSION_DENIED]
Permission was denied to access object.

[SSL_ERROR_SSL_NOT_AVAILABLE]
SSL is not available for use.

[SSL_ERROR_UNKNOWN]
An unknown or unexpected error occurred during SSL processing.

[SSL_ERROR_UNSUPPORTED_CERTIFICATE_TYPE]
i5/OS does not support the certificate’s type.

[SSL_ERROR_UNSUPPORTED_CERTIFICATE_TYPE]
i5/OS does not support the certificate’s type.

Error Conditions
When the SSL_Handshake() API fails with a return code of [SSL_ERROR_IO], errno can be set to:

[EACCES]
Permission denied.

[EBADF]
Descriptor not valid.

[EBUSY]
Resource busy.
[ECONNRESET]
A connection with a remote socket was reset by that socket.

[EDEADLK]
Resource deadlock avoided.

[EFAULT]
Bad address.
The system detected an address that was not valid while attempting to access the handle parameter or one of the address fields in the handle parameter.

[EINTR]
Interrupted function call.

[EINVAL]
Parameter not valid.
This error code indicates one of the following:
• The socket_descriptor type is not SOCK_STREAM or address family is not AF_INET or AF_INET6.
• One of the parameters passed is not valid or is NULL.
• The protocol field contains a value that is not valid.

[EALREADY]
Operation already in progress.
An SSL_Handshake() API has already been previously successfully completed.

[EIO]
Input/output error.

[ENOBUFFS]
There is not enough buffer space for the requested operation.

[ENOTCONN]
Requested operation requires a connection.
This error code indicates one of the following:
• The socket_descriptor is not for a socket that is in a connected state.
• The socket_descriptor has not had SSL support enabled.

[ENOTSOCK]
The specified descriptor does not reference a socket.

[EPipe]
Broken pipe.

[ETIMEDOUT]
A remote host did not respond within the timeout period.

[EUNATCH]
The protocol required to support the specified address family is not available at this time.

[EUNKNOWN]
Unknown system state.
Any **errno** that can be returned by **send()** or **recv()** can be returned by this API. See Sockets APIs for a description of the **errno** values they return.

If an **errno** is returned that is not in this list, see “Errno Values for UNIX-Type Functions” on page 101 for a description of the **errno**.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible PAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

**Usage Notes**

1. The **SSL_Handshake()** function is only valid on sockets that have an address family of **AF_INET** or **AF_INET6** and a socket type of **SOCK_STREAM**. If the descriptor pointed to by the **handle** structure parameter does not have the correct address family and socket type, **SSL_ERROR_IO** is returned and the **errno** value is set to **EINVAL**.

2. The **SSL_Handshake()** function can be called only one time per SSL session. If a secondary call of **SSL_Handshake()** occurs within the same established SSL session, then it will fail and the **errno** will be set to **EINVAL**.

3. A successful **SSL_Init()** or **SSL_Init_Application()** API and a successful **SSL_Create()** API must be called prior to an **SSL_Handshake()** API. The **SSL_Init()** API or **SSL_Init_Application()** API is used to establish a certificate and private key for either of the following:

   - A successful handshake as a server
   - A successful handshake as a client when connected to a server performing client authentication

4. The **SSL_Create()** API is used to enable SSL support for the specified socket descriptor.

5. When doing **SSL_Handshake()** with a **how** parameter value of **SSL_HANDSHAKE_AS_SERVER**, **SSL_HANDSHAKE_AS_SERVER_WITH_CLIENT_AUTH**, or **SSL_HANDSHAKE_AS_SERVER_WITH_OPTIONAL_CLIENT_AUTH**, the **cipherSuite** value (if **TLS_VERSION_1** or **SSL_VERSION_3** protocol) or the **cipherKind** (if **SSL_VERSION_2** protocol) will be the first cipher found in the ordered **cipherSuiteList** list that was also found in the cipher list provided by the client during the SSL handshake.

6. When doing **SSL_Handshake()** with a **how** parameter value of **SSL_HANDSHAKE_AS_CLIENT**, the cipher specification list will be sent to the server in the client hello in the order found in the **cipherSuiteList**, however the value from that list that is negotiated for the **cipherSuite** value (if **TLS_VERSION_1** or **SSL_VERSION_3** protocol) or the **cipherKind** (if **SSL_VERSION_2** protocol) is determined by the server policy.

**Related Information**

- “**SSL_Create()**—Enable SSL Support for the Specified Socket Descriptor” on page 68  — Enable SSL Support for the Specified Socket Descriptor
- “**SSL_Destroy()**—End SSL Support for the Specified SSL Session” on page 71  — End SSL Support for the Specified SSL Session
- “**SSL_Init()**—Initialize the Current Job for SSL” on page 79  — Initialize the Current Job for SSL
- “**SSL_Read()**—Receive Data from an SSL-Enabled Socket Descriptor” on page 89  — Receive Data from an SSL-Enabled Socket Descriptor
SSL_Init()—Initialize the Current Job for SSL

Syntax
#include <qsossl.h>

int SSL_Init(SSLInit* init)

Service Program Name: QSOSSLR
Default Public Authority: *USE
Threadsafe: Yes

The SSL_Init() function is used to establish the SSL security information to be used for all SSL sessions for the current job. The SSL_Init() API establishes the certificate and the associated public and private key information for use by the SSL handshake protocol processing when acting as a server or when acting as a client. The certificate and key information is needed by an application that is acting as a client in the situations where the client is connecting to a server which has enabled and requires client authentication.

Parameters
SSLInit* init (input)
The pointer to an SSLInit structure. SSLInit is a typedef for a buffer of type struct SSLInitStr. In <qsossl.h>, struct SSLInitStr is defined as the following:

```
struct SSLInitStr {
    char* keyringFileName; /* Key ring file name */
    char* keyringPassword; /* Key ring file password */
    unsigned short int* cipherSuiteList; /* List of cipher suites */
    unsigned int cipherSuiteListLen; /* number of entries in the cipher suites list */
};
```

The fields within the SSLInit structure as pointed to by init are defined as follows:

char *keyringFileName (input)
A pointer to a null-terminated character string, identifying the path to the key database file to be used for this job’s SSL processing. The path must be a fully qualified integrated file system file name.

This parameter is assumed to be represented in the CCSID (coded character set identifier) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

See “QlgSSL_Init()—Initialize the Current Job for SSL (using NLS-enabled path name)” on page 54 for a description of supplying the keyringFileName in any CCSID.
char *keyringPassword  (input)

A pointer to a null-terminated character string, identifying the password for the key database file named in the keyringFileName field.

If this parameter's value is equal to NULL, then the SSL_Init() support will attempt to extract the key database password that has been securely stored on the system.

This parameter is assumed to be represented in the CCSID (coded character set identifier) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

unsigned short int* cipherSuiteList  (input)

A pointer to the cipher specification list to be used during the SSL handshake protocol for this job. This list is a string of concatenated cipher specification values. A cipher specification value is an unsigned short integer. Any value provided will override any values provided by a previous SSL_Init() API or SSL_Init_Application() API or the system default cipher specification list if the previous SSL_Init() API or SSL_Init_Application() API did not provide a cipher specification list. A value of NULL for this parameter indicates one of the following:

- Use the cipher specification list provided by a previous SSL_Init() API or SSL_Init_Application() API
- Use the system default cipher specification list if a previous SSL_Init() API or SSL_Init_Application() API was not done

The caller specifies the preferred order of the cipher specifications. The cipher specification values, shown here not in preferred or strength order, are defined in <qsossl.h> as the following:

```
TLS_RSA_WITH_NULL_MD5         0x0001
TLS_RSA_WITH_NULL_SHA         0x0002
TLS_RSA_EXPORT_WITH_RC4_40_MD5 0x0003
TLS_RSA_WITH_RC4_128_MD5      0x0004
TLS_RSA_WITH_RC4_128_SHA      0x0005
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 0x0006
TLS_RSA_WITH_DES_CBC_SHA      0x0009
TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x000A
TLS_RSA_WITH_AES_128_CBC_SHA  0x002F
TLS_RSA_WITH_AES_256_CBC_SHA  0x0035
TLS_RSA_WITH_RC2_CBC_128_MD5  0xFF01
TLS_RSA_WITH_DES_CBC_MD5      0xFF02
TLS_RSA_WITH_3DES_EDE_CBC_MD5 0xFF03
```

Notes:
1. The SSL_RSA_EXPORT_WITH_DES40_CBC_SHA cipher is not supported by i5/OS.
2. The default cipher suite list in preference order is as follows:

```
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_DES_CBC_MD5
TLS_RSA_WITH_3DES_EDE_CBC_MD5
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
```
unsigned int cipherSuiteListLen  (input)  
    The number of cipher suite entries specified in the list pointed to by the cipherSuiteList parameter.

Authorities
Authorization of *R (allow access to the object) to the key database file and its associated files is required.

Return Value
The SSL_Init() API returns an integer. Possible values are:

[0]  
    Successful return

[SSL_ERROR_BAD_CIPHER_SUITE]  
    A cipher suite that is not valid was specified.

[SSL_ERROR_IO]  
    An error occurred in SSL processing; check the errno value.

[SSL_ERROR_KEYPASSWORD_EXPIRED]  
    The specified key ring password has expired.

[SSL_ERROR_NO_KEYRING]  
    No key ring file was specified.

[SSL_ERROR_SSL_NOT_AVAILABLE]  
    SSL is not available for use.

[SSL_ERROR_UNKNOWN]  
    An unknown or unexpected error occurred during SSL processing.

Error Conditions
When the SSL_Init() API fails with return code [SSL_ERROR_IO], errno can be set to:

[EINVAL]  
    Parameter not valid.

[EACCES]  
    Permission denied.
    This error code indicates one of the following:
    • The keyringFileName field contains a file name to which the user is not authorized.
    • The keyringPassword value is not valid for the specified keyringFileName.

[EBADF]  
    Descriptor not valid.
    This error code indicates one of the following:
    • The keyringFileName value does not specify a valid key ring file name.

[EFAULT]  
    Bad address.
The system detected an address that was not valid while attempting to access the *init* parameter or one of the address fields in the *init* parameter.

**[EUNATCH]**

The protocol required to support the specified address family is not available at this time.

**[EUNKNOWN]**

Unknown system state.

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

### Usage Notes

1. A successful *SSL_Init*, *QlgSSL_Init* (using NLS-enabled path name), or an *SSL_Init_Application()* API must be used to enable a job for SSL processing before attempting to use any other SSL API.
2. If multiple *SSL_Init_Application()* , *QlgSSL_Init*, or *SSL_Init()* APIs are performed in a job, then only the values associated with the last *SSL_Init_Application()* , *QlgSSL_Init*, or *SSL_Init()* performed are used.
3. If the *keyringPassword* parameter pointer value is equal to NULL, then *SSL_Init()* will attempt to extract the password value as stored on the system with the *keyringFileName* parameter’s value. The existence of the securely stored key database password is based on a configuration selection made during the creation of the key database file.

### Related Information

- *QlgSSL_Init()*—Initialize the Current Job for SSL (using NLS-enabled path name)
- “*SSL_Create()*—Enable SSL Support for the Specified Socket Descriptor” on page 68—Enable SSL Support for the Specified Socket Descriptor
- “*SSL_Destroy()*—End SSL Support for the Specified SSL Session” on page 71—End SSL Support for the Specified SSL Session
- “*SSL_Handshake()*—Initiate the SSL Handshake Protocol” on page 73—Initiate the SSL Handshake Protocol
- “*SSL_Init_Application()*—Initialize the Current Job for SSL Processing Based on the Application Identifier” on page 83—Initialize the Current Job for SSL Processing Based on the Application Identifier
- “*SSL_Read()*—Receive Data from an SSL-Enabled Socket Descriptor” on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- “*SSL_Write()*—Write Data to an SSL-Enabled Socket Descriptor” on page 95—Write Data to an SSL-Enabled Socket Descriptor

API introduced: V4R3
SSL_Init_Application()—Initialize the Current Job for SSL Processing Based on the Application Identifier

Syntax

#include <qsoss1.h>

int SSL_Init_Application(SSLInitApp* init_app)

Service Program Name: QSOSSLSR
Default Public Authority: *USE
Threads随处可见: Yes

The SSL_Init_Application() function is used to establish the SSL security information to be used for all SSL sessions for the current job based on the specified application identifier. The SSL_Init_Application() API uses the application identifier to determine and then establish the certificate and the associated public and private key information for use by the SSL handshake protocol processing when acting as a server or when acting as a client. The certificate and key information is needed by an application that is acting as a client in the situations where the client is connecting to a server which has enabled and requires client authentication.

Parameters

SSLInitApp * init_app (input)
   The pointer to an SSLInitApp value. SSLInitApp is a typedef for a buffer of type struct SSLInitAppStr. In <qsoss1.h>, struct SSLInitAppStr is defined as the following:

```c
struct SSLInitAppStr {
    char* applicationID; /* application id value */
    unsigned int applicationIDLen; /* length of application id */
    char* localCertificate; /* local certificate */
    unsigned int localCertificateLen; /* length of local certificate */
    unsigned short int* cipherSuiteList; /* List of cipher suites */
    unsigned int cipherSuiteListLen; /* number of entries in the cipher suites list */
    unsigned int sessionType; /* the type of application as registered */
    unsigned int reserved1; /* reserved - must be 0 */
    unsigned int protocol; /* SSL protocol version */
    unsigned int timeout; /* cache timeout (seconds) */
    char reserved[12]; /* reserved - must be NULL (0s)*/
};
```

The fields within the SSLInitApp structure as pointed to by init_app are defined as follows:

char * applicationID (input)
   A pointer to a null terminated character string identifying the application identifier value that was used to register the application using the Register Application for Certificate Use, (OPM, QSYRGAP, ILE, QsyRegisterAppForCertUse) API. See the Register Application for Certificate Use API for information on the format and values allowed for the application identifier.

char * applicationIDLen (input)
   The number of characters in the application identifier string as specified by the applicationID parameter.
char *localCertificate (input)

On input, the localCertificate pointer must be set to point to storage that has been allocated by the calling application that will be used on output to contain the application’s registered local certificate. If a certificate is not to be returned then set this pointer’s value to NULL and the localCertificateLen value to zero (0). The storage should be large enough to accommodate the size of the certificate. Most certificates are less than 2K in length. On output, the localCertificate pointer will not be changed, though the storage it points to will contain the registered application’s certificate. The certificate will be the one registered for that application by the Register Application for Certificate Use (OPM, QSYRGAP; ILE, QsyRegisterAppForCertUse) API. See the Register Application for Certificate Use API for information on the format and values allowed for the application identifier.

unsigned int localCertificateLen (input)

On input, this value must equal the number of characters available in the storage pointed to by the localCertificate pointer. Set this value to 0 if you do not want a certificate returned by this API. On output, this value is equal to the length of the certificate. If the certificate will not fit into the storage provided, then this value will be set to the length required to contain the certificate.

unsigned short int* cipherSuiteList (input)

A pointer to the cipher specification list to be used during the SSL handshake protocol for this job. This list is a string of concatenated cipher specification values. A cipher specification value is an unsigned short integer. Any value provided will override any values provided by a previous SSL_Init_Application() API or SSL_Init() API or the system default cipher specification list if the previous SSL_Init_Application() API or SSL_Init() API did not provide a cipher specification list. A value of NULL for this parameter indicates one of the following:

- Use the cipher specification list provided by a previous SSL_Init_Application() API or SSL_Init() API
- Use the system default cipher specification list if a previous SSL_Init_Application() API or SSL_Init() API was not done

The caller specifies the preferred order of the cipher specifications. The cipher specification values, shown here not in preferred or strength order, are defined in <qsossl.h> as the following:

```c
TLS_RSA_WITH_NULL_MD5         0x0001
TLS_RSA_WITH_NULL_SHA         0x0002
TLS_RSA_EXPORT_WITH_RC4_40_MD5 0x0003
TLS_RSA_WITH_RC4_128_MD5      0x0004
TLS_RSA_WITH_RC4_128_SHA      0x0005
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 0x0006
TLS_RSA_WITH_3DES_CBC_SHA     0x0009
TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x000A
TLS_RSA_WITH_AES_128_CBC_SHA  0x002F
TLS_RSA_WITH_AES_256_CBC_SHA  0x0035
TLS_RSA_WITH_RC2_CBC_128_MD5  0xFF01
TLS_RSA_WITH_DES_CBC_MD5      0xFF02
TLS_RSA_WITH_3DES_EDE_CBC_MD5 0xFF03
```

Notes:
1. The SSL_RSA_EXPORT_WITH_DES40_CBC_SHA cipher is not supported by i5/OS.
2. The default cipher suite list in preference order is as follows:

```c
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_DES_CBC_MD5
```
unsigned int cipherSuiteListLen (input)
The number of cipher suite entries specified in the list pointed to by the `cipherSuiteList` parameter.

unsigned int sessionType (output)
The type registered for the application. The following values are returned in `sessionType` and are defined in `<qsossl.h>`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SSL_REGISTERED_AS_CLIENT</td>
</tr>
<tr>
<td>1</td>
<td>SSL_REGISTERED_AS_SERVER</td>
</tr>
<tr>
<td>2</td>
<td>SSL_REGISTERED_AS_SERVER_WITH_CLIENT_AUTH</td>
</tr>
<tr>
<td>3</td>
<td>SSL_REGISTERED_AS_SERVER_WITH_OPTIONAL_CLIENT_AUTH</td>
</tr>
<tr>
<td>99</td>
<td>SSL_REGISTERED_AS_NOT_SPECIFIED</td>
</tr>
</tbody>
</table>

unsigned int reserved1 (input)
This reserved field must be set to 0.

unsigned int protocol (input)
The protocol(s) that are acceptable as the handshake protocol for this job. The following values may be specified for `protocol` and are defined in `<qsossl.h>`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SSL_VERSION_CURRENT (TLS with SSL Version 3.0 and SSL Version 2.0 compatibility)</td>
</tr>
<tr>
<td>2</td>
<td>SSL_VERSION_2 (SSL Version 2.0 only)</td>
</tr>
<tr>
<td>3</td>
<td>SSL_VERSION_3 (SSL Version 3.0 only)</td>
</tr>
<tr>
<td>4</td>
<td>TLS_VERSION_1 (TLS Version 1 only)</td>
</tr>
<tr>
<td>5</td>
<td>TLSV1_SSLV3 (TLS Version 1 with SSL Version 3.0 compatibility)</td>
</tr>
</tbody>
</table>

unsigned int timeout (input)
The time period (in seconds) for which TLS Version 1.0 and SSL Version 3.0 session parameters are cached for use with abbreviated SSL handshakes. The valid range for `timeout` is from 1 to 86,400 seconds (24 hours). Not specifying a value (0) will default to the maximum timeout, and specifying a value of 0xffffffff will disable caching. The following values are defined in `<qsossl.h>`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SSL_TIMEOUT_DEFAULT (Use default timeout, 24 hours)</td>
</tr>
<tr>
<td>86400</td>
<td>SSL_TIMEOUT_MAX (Use maximum timeout, 24 hours)</td>
</tr>
<tr>
<td>0xffffffff</td>
<td>SSL_TIMEOUT_DISABLE (Disable caching of session parameters for abbreviated handshakes)</td>
</tr>
</tbody>
</table>

char reserved[12] (input)
This reserved field must be set to NULL (0s).

**Authorities**
Authorization of *R (allow access to the object) to the key database file and its associated files is required. The certificate is stored in a key database file.

**Return Value**
The `SSL_Init_Application()` API returns an integer. Possible values are:

- **[0]**
  Successful return
- **[SSL_ERROR_BAD_CIPHER_SUITE]**
  A cipher suite that is not valid was specified.
- **[SSL_ERROR_CERT_EXPIRED]**
The validity time period of the certificate is expired.

[SSL_ERROR_KEYPASSWORD_EXPIRED]
The specified key ring password has expired.

[SSL_ERROR_NO_KEYRING]
No key ring file was found.

[SSL_ERROR_NOT_REGISTERED]
The application identifier is not registered with the certificate registry facility.

[SSL_ERROR_NOT_TRUSTED_ROOT]
The certificate is not signed by a trusted certificate authority.

[SSL_ERROR_NO_CERTIFICATE]
No certificate is available for SSL processing.

[SSL_ERROR_IO]
An error occurred in SSL processing; check the errno value.

[SSL_ERROR_SSL_NOT_AVAILABLE]
SSL is not available for use.

[SSL_ERROR_UNKNOWN]
An unknown or unexpected error occurred during SSL processing.

Error Conditions
When the SSL_Init_Application() API fails with return code [SSL_ERROR_IO], errno can be set to:

[EINVAL]
Parameter not valid.

[EACCES]
Permission denied.

This error code indicates one of the following:

• The applicationID field contains a registered application identifier to which the user is not authorized.

• The user profile, which the application is operating under, is not authorized to the key database file or its associated files.

[EFAULT]
Bad address.

The system detected an address that was not valid while attempting to access the init_app parameter or one of the address fields in the init_app parameter.

[EUNATCH]
The protocol required to support the specified address family is not available at this time.

[EUNKNOWN]
Unknown system state.
Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. Before the `SSL_Init_Application()` API can be used, the user must have registered the application using the Register Application for Certificate Use (OPM, QSYRGAP; ILE, QsyRegisterAppForCertUse) API. The Register Application For Certificate Use API registers an application with the registry facility, allowing an application to be associated with a specific certificate. The Register Application for Certificate Use is described in the System Programming Interface Reference. If the application is not registered with the registry facility, then an error of SSL_ERROR_NOT_REGISTERED will be returned by `SSL_Init_Application()`.

2. A successful `SSL_Init()`, `SSL_Init (using NLS-enabled path name)`, or an `SSL_Init_Application()` API must be used to enable a job for SSL processing before attempting to use any other SSL API.

3. If multiple `SSL_Init_Application()`, `SSL_Init (using NLS-enabled path name)`, or multiple `SSL_Init()` APIs are performed in a job, then only the values associated with the last `SSL_Init_Application()`, `SSL_Init (using NLS-enabled path name)`, or `SSL_Init()` performed are used.

4. If the `SSL_Init_Application()` API or `SSL_Init()` API are both performed in the same job, then only the values associated with the last API performed are used.

5. The reserved fields in the `SSLInitApp` structure must be set to NULLs (0s) before using this API.

Related Information

- “OlgSSL_Init”—Initialize the Current Job for SSL (using NLS-enabled path name)” on page 64—Initialize the Current Job for SSL (using NLS-enabled path name)
- “SSL_Create()—Enable SSL Support for the Specified Socket Descriptor” on page 68—Enable SSL Support for the Specified Socket Descriptor
- “SSL_Destroy()—End SSL Support for the Specified SSL Session” on page 71—End SSL Support for the Specified SSL Session
- “SSL_Init()—Initialize the Current Job for SSL” on page 79—Initialize the Current Job for SSL
- “SSL_Handshake()—Initiate the SSL Handshake Protocol” on page 73—Initiate the SSL Handshake Protocol
- “SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor” on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- “SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor” on page 95—Write Data to an SSL-Enabled Socket Descriptor

API introduced: V4R4

SSL_Perror()—Print SSL Error Message

Syntax

```
#include <qsossl.h>

void SSL_Perror(int sslreturnvalue, const char* string);
```
Service Program Name: QSOSLSSR
Default Public Authority: *USE
Threadsafe: Yes

The SSL_Perror() function prints an error message to stderr. If string is not NULL and does not point to a null character, the string pointed to by string is printed to the standard error stream. If a string is printed, it is followed by a colon and a space. Regardless of if string was printed or not, the message associated with the sslreturnvalue is printed followed by a new-line character. Also, the message associated with the thread’s errno is printed followed by a new-line character.

Parameters
int sslreturnvalue (Input)
The Return Value received from a SSL API.

char* string (Input)
The string to be printed prior to the message associated with the sslreturnvalue. If no preceding message is desired, NULL must be entered.

Authorities
No authorization is required.

Return Value
There is no return value.

Error Conditions
This API calls the Retrieve SSL Runtime Error Message (SSL_Strerror) API in order to perform its task. It inherits all error conditions from this function. If the sslreturnvalue is unrecognized or if unable to retrieve the message corresponding to sslreturnvalue, then an Unknown error message will be printed following the string. Also, the message associated with the value found in the thread’s errno is printed. Note: the value of errno may be updated by SSL_Perror() in some error conditions.

Error Messages
See Error Conditions.

Related Information
- "SSL_Strerror()—Retrieve SSL Runtime Error Message" on page 93—Retrieve SSL Runtime Error Message
- "SSL_Create()—Enable SSL Support for the Specified Socket Descriptor" on page 68—Enable SSL Support for the Specified Socket Descriptor
- "SSL_Destroy()—End SSL Support for the Specified SSL Session" on page 71—End SSL Support for the Specified SSL Session
- "SSL_Handshake()—Initiate the SSL Handshake Protocol" on page 73—Initiate the SSL Handshake Protocol
- "SSL_Init()—Initialize the Current Job for SSL" on page 79—Initialize the Current Job for SSL
- "SSL_Init_Application()—Initialize the Current Job for SSL Processing Based on the Application Identifier" on page 83—Initialize the Current Job for SSL Processing Based on the Application Identifier
- "SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor" on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- "SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor" on page 95—Write Data to an SSL-Enabled Socket Descriptor
Example
See Code disclaimer information for information pertaining to code examples.

The following example shows how **SSL_Perror()** is used:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <qosssl.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>

/*
 * bufferLen is 250 bytes */
#define bufferLen 250

void main()
{
    int bufferLen, on = 1, rc = 0, sd, sd2, addrlen = 0;
    char buffer[bufferLen];

    SSLInit sslinit;
    SSLHandle* sslh;

    struct sockaddr_in addr;

    unsigned short int cipher[3] = {
        SSL_RSA_WITH_RC4_128_MD5,
        SSL_RSA_WITH_RC4_128_SHA,
        SSL_RSA_EXPORT_WITH_RC4_40_MD5
    };

    memset((char *)&SSL_Init, 0x00, sizeof(sslinit));
    sslinit.keyringFileName = "/keyringfile.kyr";
    sslinit.keyringPassword = NULL;
    sslinit.cipherSuiteList = &cipher[0];
    sslinit.cipherSuiteListLen = 3;

    if ((rc = SSL_Init(&sslinit)) != 0)
    {
        SSL_Perror(rc, "Could not initialize SSL");
    }

    ...
}
```

API introduced: V5R1

---

**SSL_Read()**—Receive Data from an SSL-Enabled Socket Descriptor

Syntax
The SSL_Read() function is used by a program to receive data from an SSL-enabled socket descriptor.

**Parameters**

SSLHandle* handle  (input)

   The pointer to an SSLHandle for an SSL session. An SSLHandle is a typedef for a buffer of type struct SSLHandleStr. In <qsossi.h>, struct SSLHandleStr is defined as the following:

```c
struct SSLHandleStr {    /* SSLHandleStr */
   int fd;            /* Socket descriptor */
   int createFlags;  /* SSL_Create flags value */
   unsigned protocol; /* SSL protocol version */
   unsigned timeout; /* Timeout value in seconds */
   unsigned char cipherKind[3]; /* Current 2.0 cipher suite*/
   unsigned short int cipherSuite; /* Current 3.0 cipher suite */
   unsigned short int* cipherSuiteList; /* List of cipher suites */
   unsigned int cipherSuiteListLen; /* Number of entries in the cipher suites list */
   unsigned char* peerCert; /* Peer certificate */
   unsigned peerCertLen; /* Peer certificate length */
   int peerCertValidateRc; /* Return code from validation of certificate */
   int (*exitPgm)(struct SSLHandleStr* sshl);    /* Authentication exit program called when a certificate is received during SSL handshake */
};
```

void *buffer  (input)

   A pointer to the user-supplied buffer in which the data that is received on the SSL session is to be stored.

int buffer_length  (input)

   The length of the buffer.

**Authorities**

No authorization is required.

**Return Value**

The SSL_Read() API returns an integer. Possible values are:

[n]  Successful, where n is the number of bytes read.

[SSL_ERROR_BAD_MESSAGE]  SSL received a badly formatted message.
[SSL_ERROR_BAD_MAC]
A bad message authentication code was received.

[SSL_ERROR_BAD_MALLOC]
Unable to allocate storage required for SSL processing.

[SSL_ERROR_BAD_STATE]
SSL detected a bad state in the SSL session.

[SSL_ERROR_CLOSED]
The SSL session ended.

[SSL_ERROR_IO]
An error occurred in SSL processing; check the errno value.

[SSL_ERROR_PERMISSION_DENIED]
Permission was denied to access object.

[SSL_ERROR_UNKNOWN]
An unknown or unexpected error occurred during SSL processing.

[SSL_ERROR_UNSUPPORTED_CERTIFICATE_TYPE]
i5/OS does not support the certificate’s type.

**Error Conditions**

When the SSL_Read() API fails with return code [SSL_ERROR_IO], errno can be set to:

[EBADF]
Descriptor not valid.

[ECONNRESET]
A connection with a remote socket was reset by that socket.

[EFAULT]
Bad address.

One of the following conditions occurred:
- The system detected an address that was not valid while attempting to access the buffer parameter.
- The system detected an address that was not valid while attempting to access the handle parameter or one of the address fields in the handle parameter.

[EINVAL]
Parameter not valid.

This error code indicates one of the following:
- The socket descriptor type is not SOCK_STREAM or address family is not AF_INET or AF_INET6.
- One of the parameters passed is not valid or is NULL.
- The buffer_length parameter specifies a negative value.

[EIO]
Input/output error.

[ENOTCONN]
Requested operation requires a connection.

This error code indicates one of the following:
- The socket_descriptor is not for a socket that is in a connected state.
- The socket_descriptor has not had SSL support enabled. This usually means that an SSL_Create() has not been completed for this socket_descriptor.

[ENOTSOCK]  
The specified descriptor does not reference a socket.

[ETIMEDOUT]  
A remote host did not respond within the timeout period.

[EUNATCH]  
The protocol required to support the specified address family is not available at this time.

[EUNKNOWN]  
Unknown system state.

[EWOULDBLOCK]  
Operation would have caused the thread to be suspended.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. The SSL_Read() function is only valid on sockets that have an address family of AF_INET or AF_INET6 and a socket type of SOCK_STREAM. If the descriptor pointed to by the handle structure parameter does not have the correct address family and socket type, [SSL_ERROR_IO] is returned and the errno value is set to EINVAL.

2. The maximum length of data returned will not exceed 32 KB. This is due to the fact that SSL is a record level protocol and the largest record allowed is 32 KB minus the necessary SSL record headers.

3. If the createFlags field in the SSLHandle specifies a value that does not include the SSL_ENCRYPT flag, this function will simply call the sockets read() function.

4. Unpredictable results will occur when attempting to mix invocations to SSL_Read() and any of the sockets read functions (recv(), read(), readv(), and so forth). It is strongly suggested that you do not mix the SSL_Read() API with any of the sockets read functions.

5. Since SSL is a record-oriented protocol, SSL must receive an entire record before it can be decrypted and any data returned to the application. Thus, a select() may indicate that data is available to be read, but a subsequent SSL_Read() may hang waiting for the remainder of the SSL record to be received when using blocking I/O.

6. A FIONREAD ioctl() cannot be used to determine the amount of data available for reading by using SSL_Read().

7. SSL will ignore the out of band (OOB) data indicator. OOB will not affect the SSL application. OOB will just be data to the SSL protocol.
8. For an SSL enabled socket, which must use a connection-oriented transport service (that is, TCP), a returned value of zero indicates one of the following:

- The partner program has issued a close() for the socket.
- The partner program has issued a shutdown() to disable writing to the socket.
- The connection is broken and the error was returned on a previously issued socket function.
- A shutdown() to disable reading was previously done on the socket.

9. If an SSL_Read() is run on a socket that is set to non-blocking mode, and there is no data waiting to be read on the SSL enabled socket, the return value will be equal to -10 and the errno will be set to EWOULDBLOCK.

Related Information

- "SSL_Create()—Enable SSL Support for the Specified Socket Descriptor" on page 68—Enable SSL Support for the Specified Socket Descriptor
- "SSL_Destroy()—End SSL Support for the Specified SSL Session" on page 71—End SSL Support for the Specified SSL Session
- "SSL_Handshake()—Initiate the SSL Handshake Protocol" on page 73—Initiate the SSL Handshake Protocol
- "SSL_Init()—Initialize the Current Job for SSL" on page 79—Initialize the Current Job for SSL
- "SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor" on page 95—Write Data to an SSL-Enabled Socket Descriptor

API introduced: V4R3

SSL_Strerror()—Retrieve SSL Runtime Error Message

Syntax
#include <qsossl.h>

char* SSL_Strerror(int sslreturnvalue,
SSLErrorMsg* serrmsgp);

Service Program Name: QSOSSSLR
Default Public Authority: *USE
Threadsafe: Yes

The SSL_Strerror() function is used to retrieve an error message and associated text string which describes an SSL return value.

Parameters

int sslreturnvalue  (Input)
The Return Value received from a SSL API.

SSLErrorMsg* serrmsgp  (Input)
The pointer to a SSLErrorMsg structure. If no SSLErrorMsg is provided, NULL must be entered. SSLErrorMsg is a typedef for a buffer of type struct SSLErrorMsgStr. In <qsossl.h>, struct SSLErrorMsg is defined as the following:

struct SSLErrorMsgStr { /* SSLErrorMsgStr */
    char messageID[7];     /* Message identifier */
    char messageFile[20];  /* Qualified message file name */
};
The fields within the SSLErrorMsg structure as pointed to by serrmsgp are defined as follows:

char messageID[7] (output)
The message identifier which defines the message associated with the input sslreturnvalue.

char messageFile[20] (output)
The fully qualified message file name where the message associated with the messageID is stored. The first 10 characters specify the file name, and the second 10 characters specify the library.

Authorities
No authorization is required.

Return Value
The SSL_Strerror() API returns a pointer to the string. The null-terminated string is stored in the CCSID of the job. If the serrmsgp is provided, the SSLErrorMsg struct will be updated to reflect the message information corresponding to the string returned.

Error Conditions
If the sslreturnvalue is unrecognized, then an Unknown error message will be stored at the location pointed to by the return value. Other error conditions will be handled as described under Error Messages.

Error Messages
This API calls the Retrieve Message (QMHRTVM) API in order to perform its task. It inherits all error conditions from this function. If errors are encountered while using the Retrieve Message API, they will be reflected in the SSLErrorMsg fields (if provided) and any associated message replacement text will be stored at the location pointed to by the return value.

Related Information
- “SSL_Perror()—Print SSL Error Message” on page 87—Print SSL Error Message
- “SSL_Create()—Enable SSL Support for the Specified Socket Descriptor” on page 68—Enable SSL Support for the Specified Socket Descriptor
- “SSL_Destroy()—End SSL Support for the Specified SSL Session” on page 71—End SSL Support for the Specified SSL Session
- “SSL_Handshake()—Initiate the SSL Handshake Protocol” on page 73—Initiate the SSL Handshake Protocol
- “SSL_Init()—Initialize the Current Job for SSL” on page 79—Initialize the Current Job for SSL
- “SSL_Init_Application()—Initialize the Current Job for SSL Processing Based on the Application Identifier” on page 83—Initialize the Current Job for SSL Processing Based on the Application Identifier
- “SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor” on page 89—Receive Data from an SSL-Enabled Socket Descriptor
- “SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor” on page 95—Write Data to an SSL-Enabled Socket Descriptor

Example
See Code disclaimer information for information pertaining to code examples.

The following example shows how SSL_Strerror() is used:
```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <qsossl.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>

/* bufferLen is 250 bytes */
#define bufferLen 250

void main()
{
    int bufferLen, on = 1, rc = 0, sd, sd2, addrlen = 0;
    char buffer[bufferLen];

    SSLInit ssllinit;
    SSLHandle* sslh;

    struct sockaddr_in addr;

    unsigned short int cipher[3] = {
        SSL_RSA_WITH_RC4_128_MD5,
        SSL_RSA_WITH_RC4_128_SHA,
        SSL_RSA_EXPORT_WITH_RC4_40_MD5
    };

    /* memset ssllinit structure to hex zeros and */
    /* fill in values for the ssllinit structure */
    memset((char *)&SSL_Init, 0x00, sizeof(ssllinit));
    ssllinit.keyringFileName = "/keyringfile.kyr";
    ssllinit.keyringPassword = NULL;
    ssllinit.cipherSuiteList = &cipher[0];
    ssllinit.cipherSuiteListLen = 3;

    /* initialize SSL security call SSL_Init */
    if ((rc = SSL_Init(&ssllinit)) != 0) {
        printf("SSL_Init() failed with rc = %d %s \n" "and errno = %d %s \n", rc, SSL_Strerror(rc, NULL),
            errno, strerror(errno));
    }
    ...
}

API introduced: V5R1
```

---

**SSL_Write()—Write Data to an SSL-Enabled Socket Descriptor**

**Syntax**

```c
#include <qsossl.h>

int SSL_Write(SSLHandle *handle,
              void *buffer,
              int buffer_length)
```
Service Program Name: QSOSSLR
Default Public Authority: *USE
Threadsafe: Yes

The **SSL_Write()** function is used by a program to write data to an SSL-enabled socket descriptor.

### Parameters

**SSLHandle* handle** *(input)*

The pointer to an **SSLHandle** for an SSL session. An **SSLHandle** is a typedef for a buffer of type struct **SSLHandleStr**. In `<qsoss.h>`, struct **SSLHandleStr** is defined as the following:

```c
struct SSLHandleStr { /* SSLHandleStr */
    int fd;           /* Socket descriptor */
    int createFlags; /* SSL_Create flags value */
    unsigned protocol; /* SSL protocol version */
    unsigned timeout; /* Timeout value in seconds */
    unsigned char cipherKind[3]; /* Current 2.0 cipher suite*/
    unsigned short int cipherSuite; /* Current 3.0 cipher suite */
    unsigned short int* cipherSuiteList; /* List of cipher suites */
    unsigned int cipherSuiteListLen; /* Number of entries in the cipher suites list */
    unsigned char* peerCert; /* Peer certificate */
    unsigned peerCertLen; /* Peer certificate length */
    int peerCertValidateRc; /* Return code from validation of certificate */
    int (*exitPgm)(struct SSLHandleStr* sslh); /* Authentication exit program called when a certificate is received during SSL handshake */
};
```

**void *buffer** *(input)*

A pointer to the user-supplied buffer in which the data to be written is stored.

**int buffer_length** *(input)*

The length of the **buffer**.

### Authorities

No authorization is required.

### Return Value

**SSL_Write()** returns an integer. Possible values are:

*[n]*

Successful, where n is the number of bytes written.

*[SSL_ERROR_BAD_STATE]*

SSL detected a bad state in the SSL session.

*[SSL_ERROR_CLOSED]*

The SSL session ended.

*[SSL_ERROR_IO]*

An error occurred in SSL processing; check the **errno** value.

*[SSL_ERROR_UNKNOWN]*
An unknown or unexpected error occurred during SSL processing.

**Error Conditions**

When the `SSL_Write()` API fails with return code `SSL_ERROR_IO`, `errno` can be set to one of the following:

**[EBADF]**

Descriptor not valid.

**[EFAULT]**

Bad address.

One of the following conditions occurred:

- The system detected an address that was not valid while attempting to access the `buffer` parameter.
- The system detected an address that was not valid while attempting to access the `handle` parameter or one of the address fields in the `handle` parameter.

**[EINTR]**

Interrupted function call.

**[EINVAL]**

Parameter not valid.

This error code indicates one of the following:

- The `socket_descriptor` type is not `SOCK_STREAM` or address family is not `AF_INET` or `AF_INET6`.
- One of the parameters passed is not valid or is NULL.
- The `buffer_length` parameter specifies a negative value.

**[EIO]**

Input/output error.

**[ENOBUFS]**

There is not enough buffer space for the requested operation.

**[ENOTCONN]**

Requested operation requires a connection.

This error code indicates one of the following:

- The `socket_descriptor` is not for a socket that is in a connected state.
- The `socket_descriptor` has not had SSL support enabled. This usually means that an `SSL_Create()` has not been completed for this `socket_descriptor`.

**[ENOTSOCK]**

The specified descriptor does not reference a socket.

**[EPIPE]**

Broken pipe.

**[ETIMEDOUT]**

A remote host did not respond within the timeout period.

**[EUNATCH]**

The protocol required to support the specified address family is not available at this time.
Unknown system state.

Operation would have caused the thread to be suspended.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

Usage Notes

1. The `SSL_Write()` function is only valid on sockets that have an address family of `AF_INET` or `AF_INET6` and a socket type of `SOCK_STREAM`. If the descriptor pointed to by the `handle` structure parameter does not have the correct address family and socket type, `[SSL_ERROR_IO]` is returned and the `errno` value is set to `EINVAL`.

2. There is no maximum length of the data that can be written. However, SSL will segment the data into multiple SSL record buffers if it will not fit in one SSL record buffer. The maximum SSL record size is 32 KB minus the necessary SSL record headers.

3. If the `createFlags` field in the `SSLHandle` specifies a value that does not include the `SSL_ENCRYPT` flag, then this function will simply call the sockets `write()` function.

4. Unpredictable results will occur when attempting to mix calls to `SSL_Write()` and any of the sockets `write()` functions (`send()`, `write()`, `writev()`, and so forth). It is strongly suggested that you do not mix the `SSL_Write()` API with any of the sockets `write()` functions.

Related Information

- "SSL_Create()—Enable SSL Support for the Specified Socket Descriptor" on page 68—Enable SSL Support for the Specified Socket Descriptor
- "SSL_Destroy()—End SSL Support for the Specified SSL Session" on page 71—End SSL Support for the Specified SSL Session
- "SSL_Handshake()—Initiate the SSL Handshake Protocol" on page 73—Initiate the SSL Handshake Protocol
- "SSL_Init()—Initialize the Current Job for SSL" on page 79—Initialize the Current Job for SSL
- "SSL_Read()—Receive Data from an SSL-Enabled Socket Descriptor" on page 89—Receive Data from an SSL-Enabled Socket Descriptor

API introduced: V4R3
Header Files for UNIX-Type Functions

Programs using the UNIX(R)-type functions must include one or more header files that contain information needed by the functions, such as:

- Macro definitions
- Data type definitions
- Structure definitions
- Function prototypes

The header files are provided in the QSYSINC library, which is optionally installable. Make sure QSYSINC is on your system before compiling programs that use these header files. For information on installing the QSYSINC library, see [Include files and the QSYSINC Library](#).

The table below shows the file and member name in the QSYSINC library for each header file used by the UNIX-type APIs in this publication.

<table>
<thead>
<tr>
<th>Name of Header File</th>
<th>Name of File in QSYSINC</th>
<th>Name of Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>arpa/inet.h</td>
<td>ARPA</td>
<td>INET</td>
</tr>
<tr>
<td>arpa/nameser.h</td>
<td>ARPA</td>
<td>NAMESER</td>
</tr>
<tr>
<td>bse.h</td>
<td>H</td>
<td>BSE</td>
</tr>
<tr>
<td>bsedos.h</td>
<td>H</td>
<td>BSEDOS</td>
</tr>
<tr>
<td>bseerr.h</td>
<td>H</td>
<td>BSEERR</td>
</tr>
<tr>
<td>dirent.h</td>
<td>H</td>
<td>DIRENT</td>
</tr>
<tr>
<td>errno.h</td>
<td>H</td>
<td>ERRNO</td>
</tr>
<tr>
<td>fnctl.h</td>
<td>H</td>
<td>FCNTL</td>
</tr>
<tr>
<td>grp.h</td>
<td>H</td>
<td>GRP</td>
</tr>
<tr>
<td>inttypes.h</td>
<td>H</td>
<td>INTTYPES</td>
</tr>
<tr>
<td>limits.h</td>
<td>H</td>
<td>LIMITS</td>
</tr>
<tr>
<td>mman.h</td>
<td>H</td>
<td>MMAN</td>
</tr>
<tr>
<td>netdbh.h</td>
<td>H</td>
<td>NETDB</td>
</tr>
<tr>
<td>netinet/icmp6.h</td>
<td>NETINET</td>
<td>ICMP6</td>
</tr>
<tr>
<td>net/if.h</td>
<td>NET</td>
<td>IF</td>
</tr>
<tr>
<td>netinet/in.h</td>
<td>NETINET</td>
<td>IN</td>
</tr>
<tr>
<td>netinet/ip_icmp.h</td>
<td>NETINET</td>
<td>IP_ICMP</td>
</tr>
<tr>
<td>netinet/ip.h</td>
<td>NETINET</td>
<td>IP</td>
</tr>
<tr>
<td>netinet/ip6.h</td>
<td>NETINET</td>
<td>IP6</td>
</tr>
<tr>
<td>netinet/tcp.h</td>
<td>NETINET</td>
<td>TCP</td>
</tr>
<tr>
<td>netinet/udp.h</td>
<td>NETINET</td>
<td>UDP</td>
</tr>
<tr>
<td>netns/idp.h</td>
<td>NETNS</td>
<td>IDP</td>
</tr>
<tr>
<td>netns/ipx.h</td>
<td>NETNS</td>
<td>IPX</td>
</tr>
<tr>
<td>netns/ns.h</td>
<td>NETNS</td>
<td>NS</td>
</tr>
<tr>
<td>netns/sp.h</td>
<td>NETNS</td>
<td>SP</td>
</tr>
<tr>
<td>net/route.h</td>
<td>NET</td>
<td>ROUTE</td>
</tr>
<tr>
<td>nettel/tel.h</td>
<td>NETTEL</td>
<td>TEL</td>
</tr>
<tr>
<td>os2.h</td>
<td>H</td>
<td>OS2</td>
</tr>
<tr>
<td>Name of Header File</td>
<td>Name of File in QSYSINC</td>
<td>Name of Member</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>os2def.h</td>
<td>H</td>
<td>OS2DEF</td>
</tr>
<tr>
<td>pwd.h</td>
<td>H</td>
<td>PWD</td>
</tr>
<tr>
<td>Qlg.h</td>
<td>H</td>
<td>QLG</td>
</tr>
<tr>
<td>qp0lchsg.h</td>
<td>H</td>
<td>QP0LCHSG</td>
</tr>
<tr>
<td>qp0lflop.h</td>
<td>H</td>
<td>QP0LFLOP</td>
</tr>
<tr>
<td>qp0ljlrl.h</td>
<td>H</td>
<td>QP0LJRL</td>
</tr>
<tr>
<td>qp0lrro.h</td>
<td>H</td>
<td>QP0LRRO</td>
</tr>
<tr>
<td>qp0lrtsg.h</td>
<td>H</td>
<td>QP0LRTSG</td>
</tr>
<tr>
<td>qp0lsn.h</td>
<td>H</td>
<td>QP0LSCAN</td>
</tr>
<tr>
<td>Qp0lstdi.h</td>
<td>H</td>
<td>QP0LSTDJ</td>
</tr>
<tr>
<td>qp0wpid.h</td>
<td>H</td>
<td>QP0WPID</td>
</tr>
<tr>
<td>qp0zdipc.h</td>
<td>H</td>
<td>QP0ZDIPC</td>
</tr>
<tr>
<td>qp0zipc.h</td>
<td>H</td>
<td>QP0ZIPC</td>
</tr>
<tr>
<td>qp0zolip.h</td>
<td>H</td>
<td>QP0ZOLIP</td>
</tr>
<tr>
<td>qp0zolsm.h</td>
<td>H</td>
<td>QP0ZOLSM</td>
</tr>
<tr>
<td>qp0zrIPC.h</td>
<td>H</td>
<td>QP0ZRIIPC</td>
</tr>
<tr>
<td>qp0ztrc.h</td>
<td>H</td>
<td>QP0ZTRC</td>
</tr>
<tr>
<td>qp0ztrml.h</td>
<td>H</td>
<td>QP0ZTRML</td>
</tr>
<tr>
<td>qp0z1170.h</td>
<td>H</td>
<td>QP0Z1170</td>
</tr>
<tr>
<td>qsoasync.h</td>
<td>H</td>
<td>QSOASYNC</td>
</tr>
<tr>
<td>qtnxaapi.h</td>
<td>H</td>
<td>QTNXAPI</td>
</tr>
<tr>
<td>qtnxadtp.h</td>
<td>H</td>
<td>QTNXADTP</td>
</tr>
<tr>
<td>qtomeapi.h</td>
<td>H</td>
<td>QTOMEAPI</td>
</tr>
<tr>
<td>qtossapi.h</td>
<td>H</td>
<td>QT OSSAPI</td>
</tr>
<tr>
<td>resolv.h</td>
<td>H</td>
<td>RESOLVE</td>
</tr>
<tr>
<td>semaphore.h</td>
<td>H</td>
<td>SEMAPHORE</td>
</tr>
<tr>
<td>signal.h</td>
<td>H</td>
<td>SIGNAL</td>
</tr>
<tr>
<td>spawn.h</td>
<td>H</td>
<td>SPAWN</td>
</tr>
<tr>
<td>ssl.h</td>
<td>H</td>
<td>SSL</td>
</tr>
<tr>
<td>sys/errno.h</td>
<td>H</td>
<td>ERRNO</td>
</tr>
<tr>
<td>sys/ioctl.h</td>
<td>SYS</td>
<td>IOCTL</td>
</tr>
<tr>
<td>sys/ipc.h</td>
<td>SYS</td>
<td>IPC</td>
</tr>
<tr>
<td>sys/layout.h</td>
<td>H</td>
<td>LAYOUT</td>
</tr>
<tr>
<td>sys/limits.h</td>
<td>H</td>
<td>LIMITS</td>
</tr>
<tr>
<td>sys/msg.h</td>
<td>SYS</td>
<td>MSG</td>
</tr>
<tr>
<td>sys/param.h</td>
<td>SYS</td>
<td>PARAM</td>
</tr>
<tr>
<td>sys/resource.h</td>
<td>SYS</td>
<td>RESOURCE</td>
</tr>
<tr>
<td>sys/sem.h</td>
<td>SYS</td>
<td>SEM</td>
</tr>
<tr>
<td>sys/setjmp.h</td>
<td>SYS</td>
<td>SETJMP</td>
</tr>
<tr>
<td>sys/shm.h</td>
<td>SYS</td>
<td>SHM</td>
</tr>
<tr>
<td>Name of Header File</td>
<td>Name of File in QSYSINC</td>
<td>Name of Member</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>sys/signal.h</td>
<td>SYS</td>
<td>SIGNAL</td>
</tr>
<tr>
<td>sys/socket.h</td>
<td>SYS</td>
<td>SOCKET</td>
</tr>
<tr>
<td>sys/stat.h</td>
<td>SYS</td>
<td>STAT</td>
</tr>
<tr>
<td>sys/statvfs.h</td>
<td>SYS</td>
<td>STATVFS</td>
</tr>
<tr>
<td>sys/time.h</td>
<td>SYS</td>
<td>TIME</td>
</tr>
<tr>
<td>sys/types.h</td>
<td>SYS</td>
<td>TYPES</td>
</tr>
<tr>
<td>sys/uio.h</td>
<td>SYS</td>
<td>UIO</td>
</tr>
<tr>
<td>sys/un.h</td>
<td>SYS</td>
<td>UN</td>
</tr>
<tr>
<td>sys/wait.h</td>
<td>SYS</td>
<td>WAIT</td>
</tr>
<tr>
<td>ulimit.h</td>
<td>H</td>
<td>ULIMIT</td>
</tr>
<tr>
<td>unistd.h</td>
<td>H</td>
<td>UNISTD</td>
</tr>
<tr>
<td>utime.h</td>
<td>H</td>
<td>UTIME</td>
</tr>
</tbody>
</table>

You can display a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to display the `unistd.h` header file using the Source Entry Utility editor, enter the following command:
  ```
  STRSEU SRCTYPE(QSYSINC/H) SRCMBR(UNISTD) OPTION(5)
  ```

- Using the Display Physical File Member command. For example, to display the `sys/stat.h` header file, enter the following command:
  ```
  DSPPFM FILE(QSYSINC/SYS) MBR(STAT)
  ```

You can print a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to print the `unistd.h` header file using the Source Entry Utility editor, enter the following command:
  ```
  STRSEU SRCTYPE(QSYSINC/H) SRCMBR(UNISTD) OPTION(6)
  ```

- Using the Copy File command. For example, to print the `sys/stat.h` header file, enter the following command:
  ```
  CPYF FROMFILE(QSYSINC/SYS) TOFILE(*PRINT) FROMMBR(STAT)
  ```

Symbolic links to these header files are also provided in directory `/QIBM/include`.

### Errno Values for UNIX-Type Functions

Programs using the UNIX(R)-type functions may receive error information as `errno` values. The possible values returned are listed here in ascending `errno` value sequence.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Text</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDOM</td>
<td>3001</td>
<td>A domain error occurred in a math function.</td>
<td></td>
</tr>
<tr>
<td>ERANGE</td>
<td>3002</td>
<td>A range error occurred.</td>
<td></td>
</tr>
<tr>
<td>ETRUNC</td>
<td>3003</td>
<td>Data was truncated on an input, output, or update operation.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENOTOPEN</td>
<td>3004</td>
<td>File is not open.</td>
<td>You attempted to do an operation that required the file to be open.</td>
</tr>
<tr>
<td>ENOTREAD</td>
<td>3005</td>
<td>File is not opened for read operations.</td>
<td>You tried to read a file that is not open for read operations.</td>
</tr>
<tr>
<td>EINVAL</td>
<td>3008</td>
<td>Input/output error.</td>
<td>A physical I/O error occurred or a referenced object was damaged.</td>
</tr>
<tr>
<td>ENOTWRITE</td>
<td>3009</td>
<td>File is not opened for write operations.</td>
<td>You tried to update a file that has not been opened for write operations.</td>
</tr>
<tr>
<td>ESTDIN</td>
<td>3010</td>
<td>The stdin stream cannot be opened.</td>
<td></td>
</tr>
<tr>
<td>ESTDOUT</td>
<td>3011</td>
<td>The stdout stream cannot be opened.</td>
<td></td>
</tr>
<tr>
<td>ESTDERR</td>
<td>3012</td>
<td>The stderr stream cannot be opened.</td>
<td></td>
</tr>
<tr>
<td>EBADSEEK</td>
<td>3014</td>
<td>The positioning parameter in fseek is not correct.</td>
<td></td>
</tr>
<tr>
<td>EBADNAME</td>
<td>3015</td>
<td>The type variable specified on the open function is not correct.</td>
<td>The mode that you attempted to open the file in is not correct.</td>
</tr>
<tr>
<td>EBADPOS</td>
<td>3017</td>
<td>The position specifier is not correct.</td>
<td></td>
</tr>
<tr>
<td>ENOPOS</td>
<td>3018</td>
<td>There is no record at the specified position.</td>
<td>You attempted to position to a record that does not exist in the file.</td>
</tr>
<tr>
<td>ENUMRECS</td>
<td>3019</td>
<td>Attempted to use ftell on multiple members.</td>
<td>Remove all but one member from the file.</td>
</tr>
<tr>
<td>ENUMMBRS</td>
<td>3020</td>
<td>The current record position is too long for ftell.</td>
<td></td>
</tr>
<tr>
<td>EINVAL</td>
<td>3021</td>
<td>The value specified for the argument is not correct.</td>
<td>A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.</td>
</tr>
<tr>
<td>EBADFUNC</td>
<td>3022</td>
<td>Function parameter in the signal function is not set.</td>
<td></td>
</tr>
<tr>
<td>ENOENT</td>
<td>3023</td>
<td>No such path or directory.</td>
<td>The directory or a component of the path name specified does not exist.</td>
</tr>
<tr>
<td>ENOREC</td>
<td>3026</td>
<td>Record is not found.</td>
<td>You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.</td>
</tr>
<tr>
<td>EPERM</td>
<td>3027</td>
<td>The operation is not permitted.</td>
<td>You have attempted to do an operation that requires the file to be open.</td>
</tr>
<tr>
<td>EBADDATA</td>
<td>3028</td>
<td>Message data is not valid.</td>
<td>The message data that was specified for the error text is not correct.</td>
</tr>
<tr>
<td>EBUSY</td>
<td>3029</td>
<td>Resource busy.</td>
<td>An attempt was made to use a system resource that is not available at this time.</td>
</tr>
<tr>
<td>EBADOPT</td>
<td>3040</td>
<td>Option specified is not valid.</td>
<td></td>
</tr>
<tr>
<td>ENOTUPD</td>
<td>3041</td>
<td>File is not opened for update operations.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>ENOTDLT</td>
<td>3042</td>
<td>File is not opened for delete operations.</td>
<td></td>
</tr>
<tr>
<td>EPAD</td>
<td>3043</td>
<td>The number of characters written is shorter than the expected record length.</td>
<td>The length of the record is longer than the buffer size that was specified. The data written was padded to the length of the record.</td>
</tr>
<tr>
<td>EBADKEYLN</td>
<td>3044</td>
<td>A length that was not valid was specified for the key.</td>
<td>You attempted a record I/O against a keyed file. The key length that was specified is not correct.</td>
</tr>
<tr>
<td>EPUTANDGET</td>
<td>3080</td>
<td>A read operation should not immediately follow a write operation.</td>
<td></td>
</tr>
<tr>
<td>EGETANDPUT</td>
<td>3081</td>
<td>A write operation should not immediately follow a read operation.</td>
<td></td>
</tr>
<tr>
<td>EIOERROR</td>
<td>3101</td>
<td>A nonrecoverable I/O error occurred.</td>
<td></td>
</tr>
<tr>
<td>EIORECERR</td>
<td>3102</td>
<td>A recoverable I/O error occurred.</td>
<td></td>
</tr>
<tr>
<td>EACCESS</td>
<td>3401</td>
<td>Permission denied.</td>
<td>An attempt was made to access an object in a way forbidden by its object access permissions.</td>
</tr>
<tr>
<td>ENOTDIR</td>
<td>3403</td>
<td>Not a directory.</td>
<td>A component of the specified path name existed, but it was not a directory when a directory was expected.</td>
</tr>
<tr>
<td>ENOSPC</td>
<td>3404</td>
<td>No space is available.</td>
<td>The requested operations required additional space on the device and there is no space left. This could also be caused by exceeding the user profile storage limit when creating or transferring ownership of an object.</td>
</tr>
<tr>
<td>EXDEV</td>
<td>3405</td>
<td>Improper link.</td>
<td>A link to a file on another file system was attempted.</td>
</tr>
<tr>
<td>EAGAIN</td>
<td>3406</td>
<td>Operation would have caused the process to be suspended.</td>
<td></td>
</tr>
<tr>
<td>EWouldBlock</td>
<td>3406</td>
<td>Operation would have caused the process to be suspended.</td>
<td></td>
</tr>
<tr>
<td>EINTR</td>
<td>3407</td>
<td>Interrupted function call.</td>
<td></td>
</tr>
<tr>
<td>EFAULT</td>
<td>3408</td>
<td>The address used for an argument was not correct.</td>
<td>In attempting to use an argument in a call, the system detected an address that is not valid.</td>
</tr>
<tr>
<td>ETIME</td>
<td>3409</td>
<td>Operation timed out.</td>
<td></td>
</tr>
<tr>
<td>ENXIO</td>
<td>3415</td>
<td>No such device or address.</td>
<td></td>
</tr>
<tr>
<td>EAPAR</td>
<td>3418</td>
<td>Possible APAR condition or hardware failure.</td>
<td></td>
</tr>
<tr>
<td>ERECURSE</td>
<td>3419</td>
<td>Recursive attempt rejected.</td>
<td></td>
</tr>
<tr>
<td>EADDRINUSE</td>
<td>3420</td>
<td>Address already in use.</td>
<td></td>
</tr>
<tr>
<td>EADDRNOTAVAIL</td>
<td>3421</td>
<td>Address is not available.</td>
<td></td>
</tr>
<tr>
<td>EAFNOSUPPORT</td>
<td>3422</td>
<td>The type of socket is not supported in this protocol family.</td>
<td></td>
</tr>
<tr>
<td>EALREADY</td>
<td>3423</td>
<td>Operation is already in progress.</td>
<td></td>
</tr>
<tr>
<td>ECONNABORTED</td>
<td>3424</td>
<td>Connection ended abnormally.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ECONNREFUSED</td>
<td>3425</td>
<td>A remote host refused an attempted connect operation.</td>
<td></td>
</tr>
<tr>
<td>ECONNRESET</td>
<td>3426</td>
<td>A connection with a remote socket was reset by that socket.</td>
<td></td>
</tr>
<tr>
<td>EDESTADDRREQ</td>
<td>3427</td>
<td>Operation requires destination address.</td>
<td></td>
</tr>
<tr>
<td>EHOSTDOWN</td>
<td>3428</td>
<td>A remote host is not available.</td>
<td></td>
</tr>
<tr>
<td>EHOSTUNREACH</td>
<td>3429</td>
<td>A route to the remote host is not available.</td>
<td></td>
</tr>
<tr>
<td>EINPROGRESS</td>
<td>3430</td>
<td>Operation in progress.</td>
<td></td>
</tr>
<tr>
<td>EISCONN</td>
<td>3431</td>
<td>A connection has already been established.</td>
<td></td>
</tr>
<tr>
<td>EMSGSIZE</td>
<td>3432</td>
<td>Message size is out of range.</td>
<td></td>
</tr>
<tr>
<td>ENETDOWN</td>
<td>3433</td>
<td>The network currently is not available.</td>
<td></td>
</tr>
<tr>
<td>ENETRESET</td>
<td>3434</td>
<td>A socket is connected to a host that is no longer available.</td>
<td></td>
</tr>
<tr>
<td>ENETUNREACH</td>
<td>3435</td>
<td>Cannot reach the destination network.</td>
<td></td>
</tr>
<tr>
<td>ENOBUFFS</td>
<td>3436</td>
<td>There is not enough buffer space for the requested operation.</td>
<td></td>
</tr>
<tr>
<td>ENOPROTOOPT</td>
<td>3437</td>
<td>The protocol does not support the specified option.</td>
<td></td>
</tr>
<tr>
<td>ENOTCONN</td>
<td>3438</td>
<td>Requested operation requires a connection.</td>
<td></td>
</tr>
<tr>
<td>ENOTSOCK</td>
<td>3439</td>
<td>The specified descriptor does not reference a socket.</td>
<td></td>
</tr>
<tr>
<td>ENOTSUP</td>
<td>3440</td>
<td>Operation is not supported.</td>
<td></td>
</tr>
<tr>
<td>EOPNOTSUPP</td>
<td>3440</td>
<td>Operation is not supported.</td>
<td></td>
</tr>
<tr>
<td>EPFNOSUPPORT</td>
<td>3441</td>
<td>The socket protocol family is not supported.</td>
<td></td>
</tr>
<tr>
<td>EPROTONOSUPPORT</td>
<td>3442</td>
<td>No protocol of the specified type and domain exists.</td>
<td></td>
</tr>
<tr>
<td>EPROTOTYPE</td>
<td>3443</td>
<td>The socket type or protocols are not compatible.</td>
<td></td>
</tr>
<tr>
<td>ERCVDERR</td>
<td>3444</td>
<td>An error indication was sent by the peer program.</td>
<td></td>
</tr>
<tr>
<td>ESHUTDOWN</td>
<td>3445</td>
<td>Cannot send data after a shutdown.</td>
<td></td>
</tr>
<tr>
<td>ESOCKTNOSUPPORT</td>
<td>3446</td>
<td>The specified socket type is not supported.</td>
<td></td>
</tr>
<tr>
<td>ETIMEDOUT</td>
<td>3447</td>
<td>A remote host did not respond within the timeout period.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EUNATCH</td>
<td>3448</td>
<td>The protocol required to support the specified address family is not</td>
<td>A file descriptor argument was out of range, referred to a file that was not open, or a read or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>available at this time.</td>
<td>write request was made to a file that is not open for that operation.</td>
</tr>
<tr>
<td>EBADF</td>
<td>3450</td>
<td>Descriptor is not valid.</td>
<td>A file descriptor argument was out of range, referred to a file that was not open, or a read or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>write request was made to a file that is not open for that operation.</td>
</tr>
<tr>
<td>EMFILE</td>
<td>3452</td>
<td>Too many open files for this process.</td>
<td>An attempt was made to open more files than allowed by the value of OPEN_MAX. The value of OPEN_MAX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>can be retrieved using the sysconf() function.</td>
</tr>
<tr>
<td>ENFILE</td>
<td>3453</td>
<td>Too many open files in the system.</td>
<td>A system limit has been reached for the number of files that are allowed to be concurrently open</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in the system.</td>
</tr>
<tr>
<td>EPIPE</td>
<td>3455</td>
<td>Broken pipe.</td>
<td>The process that had locked the mutex is no longer running, so the mutex was deleted.</td>
</tr>
<tr>
<td>ECANCEL</td>
<td>3456</td>
<td>Operation cancelled.</td>
<td></td>
</tr>
<tr>
<td>EEXIST</td>
<td>3457</td>
<td>Object exists.</td>
<td>The object specified already exists and the specified operation requires that it not exist.</td>
</tr>
<tr>
<td>EDEADLK</td>
<td>3459</td>
<td>Resource deadlock avoided.</td>
<td>An attempt was made to lock a system resource that would have resulted in a deadlock situation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The lock was not obtained.</td>
</tr>
<tr>
<td>ENOMEM</td>
<td>3460</td>
<td>Storage allocation request failed.</td>
<td>A function needed to allocate storage, but no storage is available.</td>
</tr>
<tr>
<td>EOWNERTERM</td>
<td>3462</td>
<td>The synchronization object no longer exists because the owner is no</td>
<td>The process that had locked the mutex is no longer running, so the mutex was deleted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>longer running.</td>
<td></td>
</tr>
<tr>
<td>EDESTROYED</td>
<td>3463</td>
<td>The synchronization object was destroyed, or the object no longer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>exists.</td>
<td></td>
</tr>
<tr>
<td>ETERM</td>
<td>3464</td>
<td>Operation was terminated.</td>
<td></td>
</tr>
<tr>
<td>ENOENT1</td>
<td>3465</td>
<td>No such file or directory.</td>
<td>A component of a specified path name did not exist, or the path name was an empty string.</td>
</tr>
<tr>
<td>ENOEQFLOG</td>
<td>3466</td>
<td>Object is already linked to a dead directory.</td>
<td>The link as a dead option was specified, but the object is already marked as dead. Only one dead</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>link is allowed for an object.</td>
</tr>
<tr>
<td>EMPTYDIR</td>
<td>3467</td>
<td>Directory is empty.</td>
<td>A directory with entries of only dot and dot-dot was supplied when a nonempty directory was</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>expected.</td>
</tr>
<tr>
<td>EMLINK</td>
<td>3468</td>
<td>Maximum link count for a file was exceeded.</td>
<td>An attempt was made to have the link count of a single file exceed LINK_MAX. The value of LINK_MAX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>can be determined using the pathconf() or the fpathconf() function.</td>
</tr>
<tr>
<td>ESPPIPE</td>
<td>3469</td>
<td>Seek request is not supported for object.</td>
<td>A seek request was specified for an object that does not support seeking.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENOSYS</td>
<td>3470</td>
<td>Function not implemented.</td>
<td>An attempt was made to use a function that is not available in this implementation for any object or any arguments.</td>
</tr>
<tr>
<td>EISDIR</td>
<td>3471</td>
<td>Specified target is a directory.</td>
<td>The path specified named a directory where a file or object name was expected.</td>
</tr>
<tr>
<td>EROFS</td>
<td>3472</td>
<td>Read-only file system.</td>
<td>You have attempted an update operation in a file system that only supports read operations.</td>
</tr>
<tr>
<td>EUNKNOWN</td>
<td>3474</td>
<td>Unknown system state.</td>
<td>The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.</td>
</tr>
<tr>
<td>EITERBAD</td>
<td>3475</td>
<td>Iterator is not valid.</td>
<td></td>
</tr>
<tr>
<td>EITERSTE</td>
<td>3476</td>
<td>Iterator is in wrong state for operation.</td>
<td></td>
</tr>
<tr>
<td>EHRICLSBAD</td>
<td>3477</td>
<td>HRI class is not valid.</td>
<td></td>
</tr>
<tr>
<td>EHRICLBAD</td>
<td>3478</td>
<td>HRI subclass is not valid.</td>
<td></td>
</tr>
<tr>
<td>EHRITYPBAD</td>
<td>3479</td>
<td>HRI type is not valid.</td>
<td></td>
</tr>
<tr>
<td>ENOTAPPL</td>
<td>3480</td>
<td>Data requested is not applicable.</td>
<td></td>
</tr>
<tr>
<td>EHRIREQTyp</td>
<td>3481</td>
<td>HRI request type is not valid.</td>
<td></td>
</tr>
<tr>
<td>EHRINAMEBAD</td>
<td>3482</td>
<td>HRI resource name is not valid.</td>
<td></td>
</tr>
<tr>
<td>EDAMAGE</td>
<td>3484</td>
<td>A damaged object was encountered.</td>
<td></td>
</tr>
<tr>
<td>ELOOP</td>
<td>3485</td>
<td>A loop exists in the symbolic links.</td>
<td>This error is issued if the number of symbolic links encountered is more than POSIX_SYMLOOP (defined in the limits.h header file). Symbolic links are encountered during resolution of the directory or path name.</td>
</tr>
<tr>
<td>ENAMETOOLONG</td>
<td>3486</td>
<td>A path name is too long.</td>
<td>A path name is longer than PATH_MAX characters or some component of the name is longer than NAME_MAX characters while _POSIX_NO_TRUNC is in effect. For symbolic links, the length of the name string substituted for a symbolic link exceeds PATH_MAX. The PATH_MAX and NAME_MAX values can be determined using the <code>pathconf()</code> function.</td>
</tr>
<tr>
<td>ENOLCK</td>
<td>3487</td>
<td>No locks are available.</td>
<td>A system-imposed limit on the number of simultaneous file and record locks was reached, and no more were available at that time.</td>
</tr>
<tr>
<td>ENOTEMPTY</td>
<td>3488</td>
<td>Directory is not empty.</td>
<td>You tried to remove a directory that is not empty. A directory cannot contain objects when it is being removed.</td>
</tr>
<tr>
<td>ENOSYSRSC</td>
<td>3489</td>
<td>System resources are not available.</td>
<td>One or more characters could not be converted from the source CCSID to the target CCSID.</td>
</tr>
<tr>
<td>ECONVERT</td>
<td>3490</td>
<td>Conversion error.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>---------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>E2BIG</td>
<td>3491</td>
<td>Argument list is too long.</td>
<td></td>
</tr>
<tr>
<td>EILSEQ</td>
<td>3492</td>
<td>Conversion stopped due to input character that does not belong to the input codeset.</td>
<td></td>
</tr>
<tr>
<td>ETYPE</td>
<td>3493</td>
<td>Object type mismatch.</td>
<td>The type of the object referenced by a descriptor does not match the type specified on the interface.</td>
</tr>
<tr>
<td>EBADDIR</td>
<td>3494</td>
<td>Attempted to reference a directory that was not found or was destroyed.</td>
<td></td>
</tr>
<tr>
<td>EBADOBJ</td>
<td>3495</td>
<td>Attempted to reference an object that was not found, was destroyed, or was damaged.</td>
<td></td>
</tr>
<tr>
<td>EIDXINVAL</td>
<td>3496</td>
<td>Data space index used as a directory is not valid.</td>
<td></td>
</tr>
<tr>
<td>ESOFTDAMAGE</td>
<td>3497</td>
<td>Object has soft damage.</td>
<td></td>
</tr>
<tr>
<td>ENOTENROLL</td>
<td>3498</td>
<td>User is not enrolled in system distribution directory.</td>
<td>You attempted to use a function that requires you to be enrolled in the system distribution directory and you are not.</td>
</tr>
<tr>
<td>EOFFLINE</td>
<td>3499</td>
<td>Object is suspended.</td>
<td>You have attempted to use an object that has had its data saved and the storage associated with it freed. An attempt to retrieve the object’s data failed. The object’s data cannot be used until it is successfully restored. The object’s data was saved and freed either by saving the object with the STG(&quot;FREE) parameter, or by calling an API.</td>
</tr>
<tr>
<td>EROOBJ</td>
<td>3500</td>
<td>Object is read-only.</td>
<td>You have attempted to update an object that can be read only.</td>
</tr>
<tr>
<td>EEAHDDSI</td>
<td>3501</td>
<td>Hard damage on extended attribute data space index.</td>
<td></td>
</tr>
<tr>
<td>EEASDDSI</td>
<td>3502</td>
<td>Soft damage on extended attribute data space index.</td>
<td></td>
</tr>
<tr>
<td>EEAHDDS</td>
<td>3503</td>
<td>Hard damage on extended attribute data space.</td>
<td></td>
</tr>
<tr>
<td>EEASDDS</td>
<td>3504</td>
<td>Soft damage on extended attribute data space.</td>
<td></td>
</tr>
<tr>
<td>Eeaduprc</td>
<td>3505</td>
<td>Duplicate extended attribute record.</td>
<td></td>
</tr>
<tr>
<td>ELOCKED</td>
<td>3506</td>
<td>Area being read from or written to is locked.</td>
<td>The read or write of an area conflicts with a lock held by another process.</td>
</tr>
<tr>
<td>EFBIG</td>
<td>3507</td>
<td>Object too large.</td>
<td>The size of the object would exceed the system allowed maximum size.</td>
</tr>
<tr>
<td>EIDRM</td>
<td>3509</td>
<td>The semaphore, shared memory, or message queue identifier is removed from the system.</td>
<td></td>
</tr>
<tr>
<td>ENOMSG</td>
<td>3510</td>
<td>The queue does not contain a message of the desired type and (msgflg logically ANDed with IPC_NOWAIT).</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EFILECVT</td>
<td>3511</td>
<td>File ID conversion of a directory failed.</td>
<td>To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible.</td>
</tr>
<tr>
<td>EBADFID</td>
<td>3512</td>
<td>A file ID could not be assigned when linking an object to a directory.</td>
<td>The file ID table is missing or damaged. To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible.</td>
</tr>
<tr>
<td>ESTALE</td>
<td>3513</td>
<td>File or object handle rejected by server.</td>
<td></td>
</tr>
<tr>
<td>ESRCH</td>
<td>3515</td>
<td>No such process.</td>
<td></td>
</tr>
</tbody>
</table>
| ENOTSIGINIT  | 3516  | Process is not enabled for signals.                                 | An attempt was made to call a signal function under one of the following conditions:  
|              |       |                                                                     | • The signal function is being called for a process that is not enabled for asynchronous signals.  
<p>|              |       |                                                                     | • The signal function is being called when the system signal controls have not been initialized. |
| ECHILD       | 3517  | No child process.                                                 |                                                                       |
| EBADH        | 3520  | Handle is not valid.                                               |                                                                       |
| ETOOMANYREFS | 3523  | The operation would have exceeded the maximum number of references allowed for a descriptor. |                                                                       |
| ENOTSAFE     | 3524  | Function is not allowed.                                          | Function is not allowed in a job that is running with multiple threads. |
| EOVERFLOW    | 3525  | Object is too large to process.                                    | The object’s data size exceeds the limit allowed by this function.    |
| EJRNDAMAGE   | 3526  | Journal is damaged.                                               | A journal or all of the journal’s attached journal receivers are damaged, or the journal sequence number has exceeded the maximum value allowed. This error occurs during operations that were attempting to send an entry to the journal. |
| EJRNINACTIVE | 3527  | Journal is inactive.                                              | The journaling state for the journal is *INACTIVE. This error occurs during operations that were attempting to send an entry to the journal. |
| EJRNRCVSPC   | 3528  | Journal space or system storage error.                            | The attached journal receiver does not have space for the entry because the storage limit has been exceeded for the system, the object, the user profile, or the group profile. This error occurs during operations that were attempting to send an entry to the journal. |
| EJNRMT       | 3529  | Journal is remote.                                               | The journal is a remote journal. Journal entries cannot be sent to a remote journal. This error occurs during operations that were attempting to send an entry to the journal. |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Text</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENEWJRNRCV</td>
<td>3530</td>
<td>New journal receiver is needed.</td>
<td>A new journal receiver must be attached to the journal before entries can be journaled. This error occurs during operations that were attempting to send an entry to the journal.</td>
</tr>
<tr>
<td>ENEWJRN</td>
<td>3531</td>
<td>New journal is needed.</td>
<td>The journal was not completely created, or an attempt to delete it did not complete successfully. This error occurs during operations that were attempting to start or end journaling, or were attempting to send an entry to the journal.</td>
</tr>
<tr>
<td>EJOURNALED</td>
<td>3532</td>
<td>Object already journaled.</td>
<td>A start journaling operation was attempted on an object that is already being journaled.</td>
</tr>
<tr>
<td>EJRNENTTOOLONG</td>
<td>3533</td>
<td>Entry is too large to send.</td>
<td>The journal entry generated by this operation is too large to send to the journal.</td>
</tr>
<tr>
<td>EDATALINK</td>
<td>3534</td>
<td>Object is a datalink object.</td>
<td></td>
</tr>
<tr>
<td>ENOTAVAIL</td>
<td>3535</td>
<td>Independent Auxiliary Storage Pool (ASP) is not available.</td>
<td>The independent ASP is in Vary Configuration (VRYCFG) or Reclaim Storage (RCLSTG) processing. To recover from this error, wait until processing has completed for the independent ASP.</td>
</tr>
<tr>
<td>ENOTTY</td>
<td>3536</td>
<td>I/O control operation is not appropriate.</td>
<td></td>
</tr>
<tr>
<td>EFBIG2</td>
<td>3540</td>
<td>Attempt to write or truncate file past its sort file size limit.</td>
<td>An attempt was made to execute an i5/OS PASE program that is currently open for writing, or an attempt has been made to open for writing an i5/OS PASE program that is being executed.</td>
</tr>
<tr>
<td>ETXTBSY</td>
<td>3543</td>
<td>Text file busy.</td>
<td></td>
</tr>
<tr>
<td>EASPGRPNOTSET</td>
<td>3544</td>
<td>ASP group not set for thread.</td>
<td></td>
</tr>
<tr>
<td>ERESTART</td>
<td>3545</td>
<td>A system call was interrupted and may be restarted.</td>
<td>An object has been marked as a scan failure due to processing by an exit program associated with the scan-related integrated file system exit points.</td>
</tr>
<tr>
<td>ESCANFAILURE</td>
<td>3546</td>
<td>Object had scan failure.</td>
<td></td>
</tr>
</tbody>
</table>
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:
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:

(C) IBM 2006. Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1998, 2006. All rights reserved.

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.