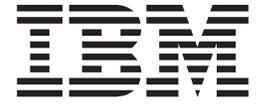


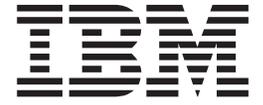
z/OS



# System Secure Sockets Layer Programming



z/OS



# System Secure Sockets Layer Programming

**Note**

Before using this information and the product it supports, be sure to read the general information under Appendix B, "Notices".

**Third Edition (September 2002)**

This document is a complete revision of SC24-5901-01.

This edition applies to Version 1 Release 4 of z/OS (program number 5694-A01), Version 1 Release 4 of z/OS.e (program number 5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. You may address your comments to the following address:

International Business Machines Corporation  
Department 55JA Mail Station P384  
2455 South Road  
Poughkeepsie, NY, 12601-5400  
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries): Your International Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrcfs@us.ibm.com

World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zosqs.html>

If you would like a reply, be sure to include your name, address, telephone number, or FAX number. Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1999, 2002. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Figures</b> . . . . .	ix
<b>Preface</b> . . . . .	xi
Who Should Use This Book . . . . .	xi
How This Book is Organized. . . . .	xi
Conventions Used in This Book . . . . .	xii
Where to Find More Information . . . . .	xiii
<b>Summary of Changes</b> . . . . .	xv
New Information for z/OS Release 4 . . . . .	xv
Changed Information for z/OS Release 4. . . . .	xvi
Deleted Information for z/OS Release 4 . . . . .	xvi
z/OS Release 2 Summary of Changes . . . . .	xvi
<b>Chapter 1. Introduction</b> . . . . .	1
Software Dependencies . . . . .	1
Installation Information . . . . .	2
<b>Chapter 2. How System SSL Works for Secure Socket Communication</b> . . . . .	5
Using System SSL on z/OS . . . . .	5
System SSL Application Overview . . . . .	6
<b>Chapter 3. Using Hardware Cryptographic Features with System SSL</b> . . . . .	9
<b>Chapter 4. Writing and Building a z/OS System SSL Application.</b> . . . . .	11
Writing a System SSL Source Program . . . . .	11
Building a z/OS System SSL Application . . . . .	16
Running a z/OS System SSL Application . . . . .	16
Additional Topics. . . . .	16
<b>Chapter 5. Migrating to the New SSL Interfaces</b> . . . . .	23
<b>Chapter 6. Building and Running a Java System SSL Application</b> . . . . .	25
Writing a Java Source Program . . . . .	25
Building a z/OS Java System SSL Application . . . . .	26
Running a z/OS Java System SSL Application . . . . .	26
<b>Chapter 7. API Reference</b> . . . . .	27
gsk_attribute_get_buffer() . . . . .	29
gsk_attribute_get_cert_info() . . . . .	31
gsk_attribute_get_data() . . . . .	34
gsk_attribute_get_enum() . . . . .	36
gsk_attribute_get_numeric_value() . . . . .	38
gsk_attribute_set_buffer() . . . . .	40
gsk_attribute_set_callback(). . . . .	43
gsk_attribute_set_enum() . . . . .	47
gsk_attribute_set_numeric_value() . . . . .	49
gsk_environment_close(). . . . .	51
gsk_environment_init() . . . . .	52
gsk_environment_open(). . . . .	54
gsk_free_cert_data() . . . . .	58
gsk_get_cert_by_label() . . . . .	59
gsk_get_cipher_suites() . . . . .	62

gsk_get_update()	63
gsk_list_free()	64
gsk_secure_socket_close()	65
gsk_secure_socket_init()	66
gsk_secure_socket_misc()	69
gsk_secure_socket_open()	71
gsk_secure_socket_read()	72
gsk_secure_socket_shutdown()	74
gsk_secure_socket_write()	76
gsk_strerror()	78
<b>Chapter 8. Certificate Management Services (CMS) API Reference</b>	<b>79</b>
gsk_add_record()	82
gsk_change_database_password()	84
gsk_change_database_record_length()	86
gsk_close_database()	87
gsk_close_directory()	88
gsk_copy_buffer()	89
gsk_copy_certificate()	90
gsk_copy_certificate_extension()	91
gsk_copy_certification_request()	92
gsk_copy_content_info()	93
gsk_copy_crl()	94
gsk_copy_name()	95
gsk_copy_private_key_info()	96
gsk_copy_public_key_info()	97
gsk_copy_record()	98
gsk_create_certification_request()	99
gsk_create_database()	101
gsk_create_self_signed_certificate()	103
gsk_create_signed_certificate()	106
gsk_create_signed_crl()	108
gsk_decode_base64()	110
gsk_decode_certificate()	111
gsk_decode_certificate_extension()	112
gsk_decode_certification_request()	114
gsk_decode_crl()	115
gsk_decode_name()	116
gsk_delete_record()	117
gsk_dn_to_name()	118
gsk_encode_base64()	120
gsk_encode_certificate_extension()	121
gsk_encode_name()	123
gsk_encode_signature()	124
gsk_export_certificate()	125
gsk_export_certification_request()	127
gsk_export_key()	129
gsk_free_buffer()	131
gsk_free_certificate()	132
gsk_free_certificates()	133
gsk_free_certificate_extension()	134
gsk_free_certification_request()	135
gsk_free_content_info()	136
gsk_free_crl()	137
gsk_free_crls()	138
gsk_free_decoded_extension()	139

gsk_free_name()	140
gsk_free_private_key_info()	141
gsk_free_public_key_info()	142
gsk_free_record()	143
gsk_free_records()	144
gsk_free_string()	145
gsk_free_strings()	146
gsk_generate_random_bytes()	147
gsk_get_cms_vector()	148
gsk_get_default_key()	149
gsk_get_default_label()	150
gsk_get_directory_certificates()	151
gsk_get_directory_crls()	152
gsk_get_record_by_id()	154
gsk_get_record_by_index()	155
gsk_get_record_by_label()	156
gsk_get_record_by_subject()	157
gsk_get_record_labels()	158
gsk_get_update_code()	159
gsk_import_certificate()	160
gsk_import_key()	162
gsk_make_content_msg()	164
gsk_make_data_content()	165
gsk_make_data_msg()	166
gsk_make_encrypted_data_content()	167
gsk_make_encrypted_data_msg()	169
gsk_make_enveloped_data_content()	171
gsk_make_enveloped_data_msg()	173
gsk_make_signed_data_content()	175
gsk_make_signed_data_msg()	177
gsk_make_wrapped_content()	179
gsk_mktime()	180
gsk_name_compare()	181
gsk_name_to_dn()	182
gsk_open_database()	183
gsk_open_database_using_stash_file()	185
gsk_open_directory()	187
gsk_open_keyring()	188
gsk_query_crypto_level()	189
gsk_query_database_label()	190
gsk_query_database_record_length()	191
gsk_rdttime()	192
gsk_read_content_msg()	193
gsk_read_data_content()	194
gsk_read_data_msg()	195
gsk_read_encrypted_data_content()	196
gsk_read_encrypted_data_msg()	198
gsk_read_enveloped_data_content()	200
gsk_read_enveloped_data_msg()	202
gsk_read_signed_data_content()	204
gsk_read_signed_data_msg()	206
gsk_read_wrapped_content()	208
gsk_receive_certificate()	209
gsk_replace_record()	210
gsk_set_default_key()	212
gsk_sign_certificate()	213

gsk_sign_crl()	214
gsk_sign_data()	215
gsk_validate_certificate()	217
gsk_verify_certificate_signature()	220
gsk_verify_crl_signature()	221
gsk_verify_data_signature()	222
<b>  Chapter 9. Deprecated Secure Sockets Layer APIs</b>	<b>225</b>
gsk_free_memory()	226
gsk_get_cipher_info()	227
gsk_get_dn_by_label()	228
gsk_initialize()	229
gsk_secure_soc_close	232
gsk_secure_soc_init()	233
gsk_secure_soc_read()	238
gsk_secure_soc_reset()	240
gsk_secure_soc_write()	241
gsk_srb_initialize()	243
GSKSRBRD	244
GSKSRBWT	245
gsk_uninitialize()	246
gsk_user_set()	247
<b>  Chapter 10. Certificate/Key Management</b>	<b>249</b>
Introduction	249
The gskkyman Command	249
Setting Up the Environment to Run gskkyman	250
Key Database Files	250
How gskkyman Works	250
Interactive Mode	253
Example Tasks Performed by the gskkyman Command in Interactive Mode	259
Example Tasks Performed by the gskkyman Command in Command Mode	281
<b>  Chapter 11. SSL Started Task</b>	<b>283</b>
GSKSRVR Environment Variables	283
Configuring the SSL Started Task	283
Server Operator Commands	284
Sysplex Session Cache Support	285
Component Trace Support.	285
<b>  Chapter 12. Obtaining Diagnostic Information</b>	<b>287</b>
Obtaining System SSL Trace Information	287
Component Trace Support.	288
Capturing Component Trace Data	288
Displaying the Trace Data	289
<b>  Chapter 13. Messages and Codes</b>	<b>291</b>
SSL Function Return Codes	291
Deprecated SSL Function Return Codes	296
ASN.1 Status Codes (014CExxx)	302
CMS Status Codes (03353xxx)	305
SSL Started Task Messages (GSK01nnn)	311
Utility Messages (GSK00nnn)	316
<b>  Appendix A. Environment Variables</b>	<b>317</b>

<b>Appendix B. Sample C++ SSL Files</b>	325
<b>Appendix C. Sample Java SSL Files</b>	327
<b>Appendix D. Accessibility</b>	329
Using assistive technologies	329
Keyboard navigation of the user interface	329
<b>Appendix E. Notices</b>	331
Programming Interface Information	332
Trademarks	333
<b>Bibliography</b>	335
z/OS Security Server Publications	335
z/OS Cryptographic Services Publications	335
IBM C/C++ Language Publication	335
Other IBM z/OS Publications	335
<b>Index</b>	337



---

## Figures

1. Sockets Programming Model Using System SSL . . . . .	8
2. Database Menu . . . . .	253
3. Key Management Menu . . . . .	255
4. Key and Certificate Menu . . . . .	255
5. Certificate Menu . . . . .	256
6. Request Menu . . . . .	257
7. Starting Menu for gskkyman . . . . .	260
8. Creating a New Key Database . . . . .	260
9. Key Management Menu for gskkyman . . . . .	261
10. Opening an Existing Key Database File . . . . .	262
11. Key Management Menu . . . . .	262
12. Deleting an Existing Key Database. . . . .	262
13. Changing a Key Database Password . . . . .	263
14. Storing a Database Password in a Stash File . . . . .	264
15. Select 6 to Create a Self-Signed Certificate . . . . .	265
16. Creating a Self-Signed Certificate . . . . .	266
17. Select 4 to Create a New Certificate Request . . . . .	267
18. Creating a Certificate Request . . . . .	267
19. Contents of certreq.arm after Certificate Request Generation . . . . .	268
20. Receiving a Certificate Issued for your Request . . . . .	269
21. Key and Certificate List . . . . .	270
22. Key and Certificate Menu . . . . .	270
23. Certificate Information . . . . .	271
24. Certificate Extensions List . . . . .	271
25. Key Usage Information . . . . .	271
26. Key Information menu . . . . .	272
27. Marking a Certificate (and Private Key) as the Default Certificate. . . . .	272
28. Copying a Certificate Without its Private Key . . . . .	273
29. Copying a Certificate and Private key to a Different Key Database . . . . .	274
30. Copying a Certificate with its Private Key to a Key Database on the Same System . . . . .	275
31. Delete Certificate and Key . . . . .	276
32. Changing a Certificate Label . . . . .	277
33. Certificate List (part 1) . . . . .	277
34. Certificate List (part 2) . . . . .	278
35. Importing a Certificate from a File . . . . .	278
36. Importing a Certificate and Private Key from a File . . . . .	279
37. Being Your Own CA in a Web Network . . . . .	280



---

## Preface

This book contains information about the System SSL product. This information consists of primarily two sets of APIs and a Certificate Management utility. The first set of APIs support the Secure Sockets Layer protocols (SSL V2.0, SSL 3.0 and TLS V1.0) which can be utilized by C/C++ applications to communicate securely across an open communications network. The other set of APIs (Certificate Management) provide the ability to exploit function other than the SSL protocols. These functions include the ability to create/manage key database files in a similar function to the SSL Certificate Management utility, use certificates stored in the key database file or key ring for purposes other than SSL and basic PKCS #7 message support to provide application writers a mechanism to communicate with another application through the PKCS #7 standard.

This book also provides guidance on how to write a client and server secure sockets layer application. The client and server may both reside on z/OS systems or reside on different systems.

---

## Who Should Use This Book

This document is intended to assist system administrators in setting up the system to use System SSL support and application programmers in writing System SSL applications.

---

## How This Book is Organized

The following briefly describes the format and organization of this book:

Chapter 1, "Introduction" on page 1 describes Secure Sockets Layer (SSL) and lists the software dependencies and installation information you need to use the System SSL support.

Chapter 2, "How System SSL Works for Secure Socket Communication" on page 5 provides a general overview of System SSL and the basic structure of a z/OS application using System SSL.

Chapter 3, "Using Hardware Cryptographic Features with System SSL" on page 9 describes the Hardware Cryptographic features.

Chapter 4, "Writing and Building a z/OS System SSL Application" on page 11 describes how to write a System SSL source program and build the System SSL application.

Chapter 5, "Migrating to the New SSL Interfaces" on page 23 describes how to migrate your existing application programs from the deprecated SSL interfaces to the new SSL interfaces.

Chapter 6, "Building and Running a Java System SSL Application" on page 25 describes how to write a Java System SSL source program and build the application.

Chapter 7, "API Reference" on page 27 describes the System SSL program interfaces.

Chapter 8, "Certificate Management Services (CMS) API Reference" on page 79 describes the Certificate Management Services (CMS) program interfaces.

Chapter 9, "Deprecated Secure Sockets Layer APIs" on page 225 describes the deprecated System SSL program interfaces.

Chapter 10, "Certificate/Key Management" on page 249 describes how to use the gskkyman utility to create a key database file, a public/private key pair, a certificate request, and other tasks.

Chapter 11, “SSL Started Task” on page 283 provides sysplex session cache support and dynamic trace support.

| Chapter 12, “Obtaining Diagnostic Information” on page 287 provides debugging information.

| Chapter 13, “Messages and Codes” on page 291 contains various messages and codes you may encounter using System SSL.

| Appendix A, “Environment Variables” on page 317 lists the environment variables used by System SSL.

| Appendix B, “Sample C++ SSL Files” on page 325 describes the sample set of files shipped to provide an example of what is needed to build a C++ System SSL application.

| Appendix C, “Sample Java SSL Files” on page 327 describes the sample set of files shipped which provide an example of what is needed to build a z/OS System SSL Java application.

| Appendix D, “Accessibility” on page 329 describes accessibility features provided with this product to help a user who has a physical disability.

| Appendix E, “Notices” on page 331 lists various trademark and licensing notices.

---

## Conventions Used in This Book

This book uses the following typographic conventions:

**Bold**   **Bold** words or characters

### Highlighting<sup>1</sup>

Words or characters **highlighted** in this manner represent system elements that you must enter into the system literally, such as commands, options, or path names.

*Italic*   *Italic* words or characters

### Highlighting<sup>2</sup>

Words or characters *highlighted* in this manner represent values for variables that you must supply.

### Example font

Examples and information displayed by the system appear in constant width type style.

[ ]   Brackets enclose optional items in format and syntax descriptions.

{ }   Braces enclose a list from which you must choose an item in format and syntax descriptions.

|   A vertical bar separates items in a list of choices.

< >   Angle brackets enclose the name of a key on the keyboard.

...   Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

\   A backslash is used as a continuation character when entering commands from the shell that exceed one line (255 characters). If the command exceeds one line, use the backslash character \ as the last nonblank character on the line to be continued, and continue the command on the next line.

This book uses the following keying conventions:

### <ALT-c>

The notation <Alt->c followed by the name of a key indicates a control character sequence.

## <Return>

The notation **<Return>** refers to the key on your keyboard that is labeled with the word Return or Enter, or with a left arrow.

## Entering commands

When instructed to enter a command, type the command name and then press **<Return>**.

---

## Where to Find More Information

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of z/OS, see the *z/OS: Information Roadmap*, SA22-7500. For a list of titles and order numbers of the books that are useful for the SSL Services, see “Bibliography” on page 335.

## Softcopy Publications

The z/OS Cryptographic Services library is available on a CD-ROM, z/OS Collection, SK3T-4269. The CD-ROM online library collection is a set of unlicensed books for z/OS and related products that includes the IBM Library Reader. This is a program that enables you to view the BookManager files. This CD-ROM also contains the Portable Document Format (PDF) files. You can view or print these files with the Adobe Acrobat reader.

## Internet Sources

The softcopy z/OS publications are also available for web-browsing and for viewing or printing PDFs using the following URL:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/>

You can also provide comments about this book and any other z/OS documentation by visiting that URL. Your feedback is important in helping to provide the most accurate and high-quality information.

## Accessing Licensed Books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link site:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link user ID, password, and z/OS licensed book key code. The z/OS order that you received provides a memo that includes your key code.

To obtain your IBM Resource Link user ID and password, logon to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.
3. Select **Access Profile**.
4. Select **Request Access to Licensed books**.
5. Supply your key code where requested and select the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed.

After your request is processed you will receive an e-mail confirmation.

**Note:** You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. Select **Library**.
3. Select **zSeries**.
4. Select **Software**.
5. Select **z/OS**.
6. Access the licensed book by selecting the appropriate element.

## Using LookAt to Look Up Message Explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages, system abends, and some codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

or from anywhere in z/OS where you can access a TSO command line (for example, TSO prompt, ISPF, z/OS UNIX Systems Services running OMVS).

To find a message explanation on the Internet, go to the LookAt Web site and simply enter the message identifier (for example, **IAT1836** or **IAT\***). You can select a specific release to narrow your search. You can also download code from the *z/OS Collection*, SK3T-4269, and the LookAt Web site so you can access LookAt from a PalmPilot (Palm Vllx suggested).

To use LookAt as a TSO command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO from a disk on your *z/OS Collection*, SK3T-4569, or from the LookAt Web site. To obtain the code from the LookAt Web site, do the following:

1. Go to <http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>.
2. Click the **News** button.
3. Scroll to **Download LookAt Code for TSO and VM**.
4. Click the ftp link, which takes you to a list of operating systems. Select the appropriate operating system. Then select the appropriate release.
5. Find the **lookat.me** file and follow its detailed instructions.

To find a message explanation from a TSO command line, simply enter: **lookat message-id**. LookAt displays the message explanation for the message requested.

**Note:** Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS: Routing and Descriptor Codes*, SA22-7624. For such messages, LookAt prompts you to choose which book to open.

## How to Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book, send your comments by using Resource Link at

<http://www.ibm.com/servers/resourceLink>

Select Feedback on the Navigation bar on the left. Be sure to include the name of the book, the form number of the book, the version of the book, if applicable, and the specific location of the text you are commenting on (for example, a page number or table number.)

---

# Summary of Changes

## Summary of Changes for SC24-5901-02 z/OS Version 1 Release 4

This book contains information previously presented in *z/OS System Secure Sockets Layer Programming, SC24-5901-01*, which supports z/OS Version 1 Release 2.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this book--for example, procedures that have a different look and format. These changes are ongoing improvements to the consistency and retrievability of information in our books.

The following summarizes the technical changes to the book:

---

## New Information for z/OS Release 4

- gskkyman utility has been restructured to allow for clearer presentation of certificate information as well as enhanced to support exporting/importing certificates in PKCS #12 Version 3 and PKCS #7 format, modification of certificate labels and creation of Digital Signature Standard certificates (FIPS 186-1).
- New Certificate Management APIs  
In addition to the APIs being provided so that applications can securely communicate over an open communication network using the SSL or TLS protocols, a new suite of APIs has been introduced to allow application writers the ability to exploit function other than the typical SSL functions. These functions include
  - The ability to create/manage key database files in a similar function to the SSL gskkyman utility.
  - Use certificates stored in the key database file or key ring for purposes other than SSL.
  - Basic PKCS #7 message support has been added to provide application writers a mechanism to communicate with another application through the PKCS #7 standard. These APIs build and process the PKCS #7 messages.
- External Security Manager (i.e. RACF) key ring support has been enhanced to allow private keys to be stored in ICSF and applications to use key rings owned by other userids.
- Added AES cipher support to its SSL V3.0 and TLS V1.0 implementations. In order to exploit the AES ciphers, Security Level 3 Feature of System SSL is required.
- Support for IPv6 network addresses has been added.
- Performance enhancements  
An in storage caching mechanism has been added where retrieved Certificate Revocation Lists (CRLs) will be cached for a period of time. This will optimize the fetching done to retrieve CRL information from the LDAP server during certificate validation.  
A Sysplex session cache has been added to make SSL server session information available across the sysplex. An SSL session established with a server on one system in the sysplex can be resumed using a server on another system in the sysplex as long as the SSL client presents the session identifier obtained for the first session when initiating the second session. The sysplex session cache can be used to store SSL V3.0 and TLS V1.0 server session information.
- Serviceability  
Component trace and enhanced debug granularity of trace information has been added.
- Messages and Codes  
Explanations and actions have been added for the System SSL API return codes and utility messages.

---

## Changed Information for z/OS Release 4

- When creating certificates, the Key Usage extension identifies how the certificate can be used. In prior releases, this extension was never created. In V1R4, new certificates will take advantage of this extension. When creating a certificate to be used by a client or server application, the certificate must be created as an enduser certificate. This will allow it to do key encipherment and digital signature. When the certificate is to act as a signing certificate authority, it must be created as a CA certificate. This will update the Key Usage with CRL and signing capabilities.
- **gsk\_attribute\_get\_cert\_info()** has been enhanced to return additional fields from the certificate (i.e., subject alternate name values)
- **gsk\_environment\_close()** will allow you to close an environment with active connections. The connections will be allowed to continue to completion.
- **gsk\_secure\_soc\_init()** will fail if the specified DN is not unique within the key database.
- **gsk\_get\_update()** has been enhanced to let an application query whether the key database file has been modified since the SSL environment was established.
- Deprecated APIs updated to support the TLS V1.0 protocol.

---

## Deleted Information for z/OS Release 4

- Certificate creation through gskkyman will support certificates with key sizes of 1024 and 2028. Existing certificates with key sizes of 512 will continue to be supported as well as the importation of certificates with key sizes of 512.
- Certificate creation through gskkyman will support X.509 Version 3 certificates only. Existing Version 1 and Version 2 certificates will continue to be supported as well as the importation of certificates based on version 1 and 2.
- gskkyman will no longer support the migration of mkkf certificate files to key database files.

---

## z/OS Release 2 Summary of Changes

### Summary of Changes for SC24-5901-01 z/OS Version 1 Release 2

This book contains information previously presented in *z/OS System Secure Sockets Layer Programming, SC24-5901-00*, which supports z/OS Version 1 Release 1.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

The following summarizes the technical changes to the book:

- An appendix with z/OS product accessibility information has been added.

## New Information for z/OS Release 2

- Added support for Transport Layer Security (TLS) V1.0 protocol (RFC 2246)
- New set of APIs (APIs created prior to this release of System SSL are deprecated, but are still supported for use by existing application programs). The new APIs allow for:
  - Greater flexibility in defining the SSL environment and sessions, and
  - The capability of defining multiple concurrent SSL environments, to allow SSL environment attributes to be modified without disrupting any SSL sessions already in progress.

- Addition of support to allow the LDAP server containing information about Certificate Revocations Lists to be recycled after the SSL environment has been defined.
- Addition of support for Certificate Revocation Lists created through the Tivoli PKI Trust Authority.
- Addition of information on usage of hardware cryptographic features by System SSL.
- Addition of enhanced diagnostic information. This includes how to enable internal tracing and use of the **gsktrace** trace formatting tool.

## **Deleted Information for z/OS Release 2**

- The sample C++ SSL files were removed from Appendix B, "Sample C++ SSL Files" on page 325. These files are shipped in `/usr/lpp/gskssl/examples`.
- The sample Java SSL files were removed from Appendix C, "Sample Java SSL Files" on page 327. These files are shipped in `/usr/lpp/gskssl/examples/java`.



---

# Chapter 1. Introduction

Secure Sockets Layer (SSL) is a communications protocol that provides secure communications over an open communications network (for example, the Internet). The SSL protocol is a layered protocol that is intended to be used on top of a reliable transport, such as Transmission Control Protocol (TCP/IP). SSL provides data privacy and integrity as well as server and client authentication based on public key certificates. Once an SSL connection is established between a client and server, data communications between client and server are transparent to the encryption and integrity added by the SSL protocol. System SSL supports the SSL V2.0, SSL V3.0 and TLS (Transport Layer Security) V1.0 protocols. TLS V1.0 is the latest version of the secure sockets layer protocol.

**Note:** Throughout this book, the phrase SSL is used to describe both the SSL and TLS protocols.

z/OS provides a set of SSL C/C++ callable application programming interfaces that, when used with the z/OS Sockets APIs, provide the functions required for applications to establish this secure sockets communications.

In addition to providing the API interfaces to exploit the Secure Sockets Layer and Transport Layer Security protocols, System SSL is also providing a suite of Certificate Management APIs. These APIs give the capability to create/manage your own certificate databases, utilize certificates stored in key database and key rings for purpose other than SSL and to build/process PKCS #7 standard messages.

---

## Software Dependencies

- Cryptographic Services System SSL (Function Modification Identifier (FMID) HCPT340)  
System SSL Version 1 Release 4 is part of System SSL Cryptographic Services Base element of z/OS. (The System SSL Cryptographic Services Base members are installed in the *pdsname.SGSKLOAD* Partitioned Data Set, or PDS and *pdsname.SGSKSAMP* PDS.)
- Cryptographic Services Security Level 3 (FMID JCPT341)  
When you order the Cryptographic Services Security Level 3 support, GSKSUS31 is installed as a member of the *pdsname.SGSKLOAD* PDS. *pdsname.SGSKLOAD* is the PDS in which the System SSL Cryptographic Services Base members are installed.
- Japanese (FMID JCPT34J)  
Contains Japanese message text files for gskkyman utility. The *gskmsgs.cat* file is installed in the */usr/lpp/gskssl/lib/nls/msg/Ja\_JP.IBM-939* directory.

The following table lists the encryption capabilities for each FMID:

Table 1. FMID Encryption Capabilities

Encryption Types/Key Sizes	Base Security Level FMID HCPT340	Security Level 3 FMID JCPT341
512 bit keys	X	X
1024 bit keys	X	X
2048 bit keys	X	X
1 - SSL V2.0 RC4 US		X
2 - SSL V2.0 RC4 Export	X	X
3 - SSL V2.0 RC2 US		X
4 - SSL V2.0 RC2 Export	X	X
6 - SSL V2.0 DES 56-Bit	X	X
7 - SSL V2.0 Triple DES US		X

Table 1. FMID Encryption Capabilities (continued)

Encryption Types/Key Sizes	Base Security Level FMID HCPT340	Security Level 3 FMID JCPT341
01 - SSL V3.0 NULL MD5	X	X
02 - SSL V3.0 NULL SHA-1	X	X
03 - SSL V3.0 RC4 MD5 Export	X	X
04 - SSL V3.0 RC4 MD5 US		X
05 - SSL V3.0 RC4 SHA-1 US		X
06 - SSL V3.0 RC2 MD5 Export	X	X
09 - SSL V3.0 DES SHA-1 Export	X	X
0A - SSL V3.0 Triple DES SHA-1 US		X
2F - SSL V3.0 AES 128 Bit SHA-1		X
35 - SSL V3.0 AES 256-Bit SHA-1		X

## Installation Information

System SSL is part of the System SSL Cryptographic Services Base element of z/OS. If you choose to install the z/OS Release 4 Server Pack, you will not need to install the System SSL Cryptographic Services Base element separately. If you choose the z/OS PDO, you can install the System SSL Cryptographic Services Base element using SMP/E. The *z/OS: Program Directory*, GI10-0669 contains the directions for installing the System SSL Cryptographic Services Base element using SMP/E.

## System SSL Parts Shipped in HFS

- /usr/lpp/gskssl/include  
Contains the header files, **gskssl.h**, **gsktypes.h** and **gskcms.h**, which declare structures and constants that are used by the System SSL and Certificate Management interfaces .
- /usr/lpp/gskssl/examples  
Contains sample client/server files as well as a display\_certificate sample program.
- /usr/lpp/gskssl/lib  
Contains GSKSSL.x file, which provides access to the APIs that are exported by the GSKSSL DLL and GSKCMS31.x which provides access to the APIs that are exported by the GSKCMS31 DLL. You use GSKSSL.x and GSKCMS31.x when you linkedit your source program that uses System SSL. This directory also contains the libSSLSocketLib.so file that provides System SSL functionality to the Java class files.
- /usr/lpp/gskssl/lib/nls/msg/En\_US.IBM-1047  
Contains the gskmsgs.cat message catalog file.
- /usr/lpp/gskssl/bin  
Contains the gskkyman and gsktrace utilities.
- /usr/lpp/gskssl/examples/java  
Contains the example Java client and server.
- /usr/lpp/gskssl/java  
Contains the GSKSSL.jar file that allows users to access System SSL functionality through Java.
- /usr/lpp/gskssl/java/docs  
Contains the javadoc api information for the Java System SSL functionality.

## System SSL Parts Shipped in PDS

| *pdsname*.SGSKLOAD PDS contains members **GSKSSL**, **GSKCMS31**, **GSKSCTFT**, **GSKSCTSS**,  
| **GSKSRBRD**, **GSKSRBWT** and **GSKSRVR** when the base FMID HCPT340 is installed. When JCPT341 is  
| installed, member **GSKSUS31** is also in the PDS.

| *pdsname*.SGSKSAMP PDS contains members **GSKMSGXT**, **GSKSRVR** and **GSKWTR**.

*pdsname* is the name determined during installation. You will need to know the name of this PDS when you identify the STEPLIB in the runtime steps. Refer to the *z/OS: Program Directory*, GI10-0669 for information about installing the System SSL.

**Note:** The DLLs are shipped in PDS form so the DLLs can be called from HFS-based as well as PDS-based programs.

**Note:** The DLLs are not placed in LPALIB or LINKLIB by default during installation. The system administrator can place the DLLs in LPALIB or LINKLIB after installation, if desired.



---

## Chapter 2. How System SSL Works for Secure Socket Communication

System SSL supports both the TLS (Transport Layer Security) and SSL (Secure Sockets Layer) protocols. Before you start writing your application, let's look at how System SSL works.

**Note:** Throughout this book, the phrase *SSL* is used to describe both the SSL and TLS protocols.

The SSL protocol begins with a "handshake". During the handshake, the client authenticates the server, the server optionally authenticates the client and the client and server agree on how to encrypt and decrypt information. In addition to the "handshake", SSL also defines the format used to transmit encrypted data.

X.509 certificates are used by both the client and server when securing communications using System SSL. The client must verify the server's certificate based on the certificate of the Certificate Authority (CA) that signed the certificate or based on a self-signed certificate from the server. The server must verify the client's certificate (if requested) using the certificate of the CA that signed the client's certificate. The client and the server then use the negotiated session keys and begin encrypted communications.

The SSL protocol runs above the TCP/IP and below higher-level protocols such as HTTP. It uses TCP/IP on behalf of the higher-level protocols.

The capabilities of SSL address several fundamental concerns about communication over the Internet and other TCP/IP networks:

**SSL server authentication** allows a client application to confirm the identity of the server application. The client application through SSL uses standard public-key cryptography to verify that the server's certificate and public key are valid and has been signed by a trusted certificate authority (CA) that is known to the client application.

**SSL client authentication** allows a server application to confirm the identity of the client application. The server application through SSL uses standard public-key cryptography to verify the the client's certificate and public key are valid and has been signed by a trusted certificate authority (CA ) that is known to the server application.

**An encrypted SSL connection** requires all information being sent between the client and server application to be encrypted. The sending application is responsible for encrypting the data and the receiving application is responsible for decrypting the data. In addition to encrypting the data, SSL provides message integrity. Message integrity provides a means to determine if the data has been tampered with since it was sent by the partner application.

---

### Using System SSL on z/OS

System SSL provides programming interfaces to write both client and server applications. These programming interfaces provide functionality associated with either the SSL environment layer or secure socket connection layer. The SSL environment layer defines the general attributes of the environment, such as the key database file name, stash file name and session timeout. The secure socket connection layer defines the attributes associated with each secure connection being established, such as the file descriptor and certificate label. The SSL application program must first create the SSL environment layer. Once the environment is created, one or more instances of the secure socket connection layer can be associated with the SSL environment. Each of these secure socket connections can be established and closed independently of each other.

Each layer has four general function calls:

## How System SSL Works

- open
- attribute\_set
- initialize
- close

In addition, the secure socket connection layer has read and write function calls for reading and writing secure data between the two SSL enabled applications.

The open function calls return a handle (an environment handle or a secure socket connection handle) that must be passed back as a parameter on subsequent function calls. An instance of a secure socket connection handle is associated with an environment by passing the environment handle as a parameter on the **gsk\_secure\_socket\_open()** call. The **gsk\_secure\_socket\_open()** function is completely thread safe. Invocations to the **gsk\_secure\_socket\_open()** function can be issued from different threads within an environment. Read and write functions are full-duplex, so asynchronous read and write function calls can be performed from different threads for a given secure socket connection. However, there can only be one read and one write call in progress at one time for any secure socket connection handle.

For every *open*, there must be a corresponding *close*.

In addition to the above functions, various **gsk\_attribute\_set ...()** and **gsk\_attribute\_get...()** functions exist to define and retrieve attributes values associated with either the environment or secure socket connection layers. The syntax of these function calls is the same for both the environment and the secure socket connection layers. The target for the set/get function is determined by the handle specified on the function call.

---

## System SSL Application Overview

Figure 1 on page 8 describes the basic structure of the elements needed in your System SSL source program.

Whether writing a server or client applications, the initial steps are the same. First, an SSL environment must be established with the following function calls:

### **gsk\_environment\_open()**

This is the first function call. It returns an environment handle that is used in all subsequent function calls. It also obtains storage and sets default values for all internal variables and picks up the values specified in system environment variables that override the built-in defaults.

### **gsk\_attribute\_set...()**

One or more of these function calls are issued to set attribute values for the environment.

### **gsk\_environment\_init()**

After you have set all variables, issue this function call to complete the initialization of the SSL environment. Once you have done this, you can open and close SSL connections.

At this point the client and server sides diverge. The server side sets up a listen environment. The listen environment is established by obtaining a socket descriptor through the **socket()** call and the activation of a connection through the **bind()**, **listen()** and **accept()** socket calls. Once the listen environment is established, the server waits for notification that a secure socket connection is requested and issues the following System SSL API function calls:

### **gsk\_secure\_socket\_open()**

This function call reserves a handle in which to store information for initializing each secure socket. Default values for each SSL connection are set from the environment.

### **gsk\_attribute\_set...()**

These function calls set attribute values for this particular SSL connection. These values could include the socket file descriptor, ciphers, protocol, and application-supplied callback routines.

### **gsk\_secure\_socket\_init()**

For each connection to be started, the application must issue this function call to complete the initialization of the SSL connection and to run the SSL handshake protocol. The SSL handshake is a function of the system SSL support.

### **gsk\_secure\_socket\_read()**

One or more read function calls is issued until the inbound data flow is complete. The number of calls is purely application-dependent.

### **gsk\_secure\_socket\_write()**

One or more write function calls is issued until all appropriate data is sent to the partner. Reads and writes may be alternated as defined by the application protocol until the data flow is complete.

### **gsk\_secure\_socket\_close()**

This function call frees all the resources used for the SSL connection.

All of the SSL API function calls are thread-safe. This is particularly useful on the server side, since each connection can be run on its own thread, simplifying application design. See the sample client/server program shipped with z/OS System SSL, for an illustration of multi-threaded application.

The client application then opens a connection to the server through the **socket()** and **connect()** calls and issues the following System SSL API function calls:

### **gsk\_secure\_socket\_open()**

This function call reserves a handle in which to store information for initializing each secure socket.

### **gsk\_attribute\_set...()**

These function calls set values for this particular SSL connection. These values could include the socket file descriptor, ciphers, protocol, and application-supplied callback routines.

### **gsk\_secure\_socket\_init()**

For each connection to be started, the application must issue this function call to complete the initialization of the SSL connection and to run the SSL handshake protocol. The SSL handshake is a function of the System SSL support.

### **gsk\_secure\_socket\_write()**

One or more write function calls are issued until the outbound data flow is complete. The number of calls is purely application-dependent.

### **gsk\_secure\_socket\_read()**

One or more read function calls are issued until all appropriate data is received from the partner. Writes and reads may be alternated as defined by the application protocol until the data flow is complete.

### **gsk\_secure\_socket\_close()**

This function call frees all the resources used for the SSL connection.

For both client and server applications, when the application is ready to end and all **gsk\_secure\_socket\_close()** functions have completed, destroy the sockets through the **close()** call and issue the **gsk\_environment\_close()** function call to close the SSL environment and return resources to the operating system.

**Note:** **skread** and **skwrite** are the routines responsible for sending and receiving data from the socket. They are invoked by the **gsk\_secure\_socket\_init()**, **gsk\_secure\_socket\_read()** and **gsk\_secure\_socket\_write()** functions.

In addition to using the previous SSL programming interfaces in an application, an application is not complete until a key database is available for use by the SSL application. The key database contains certificate information and can be an HFS file built and managed using the **gskkyman** utility or a RACF

## How System SSL Works

key ring. For more information about key databases, refer to Chapter 10, "Certificate/Key Management" on page 249.

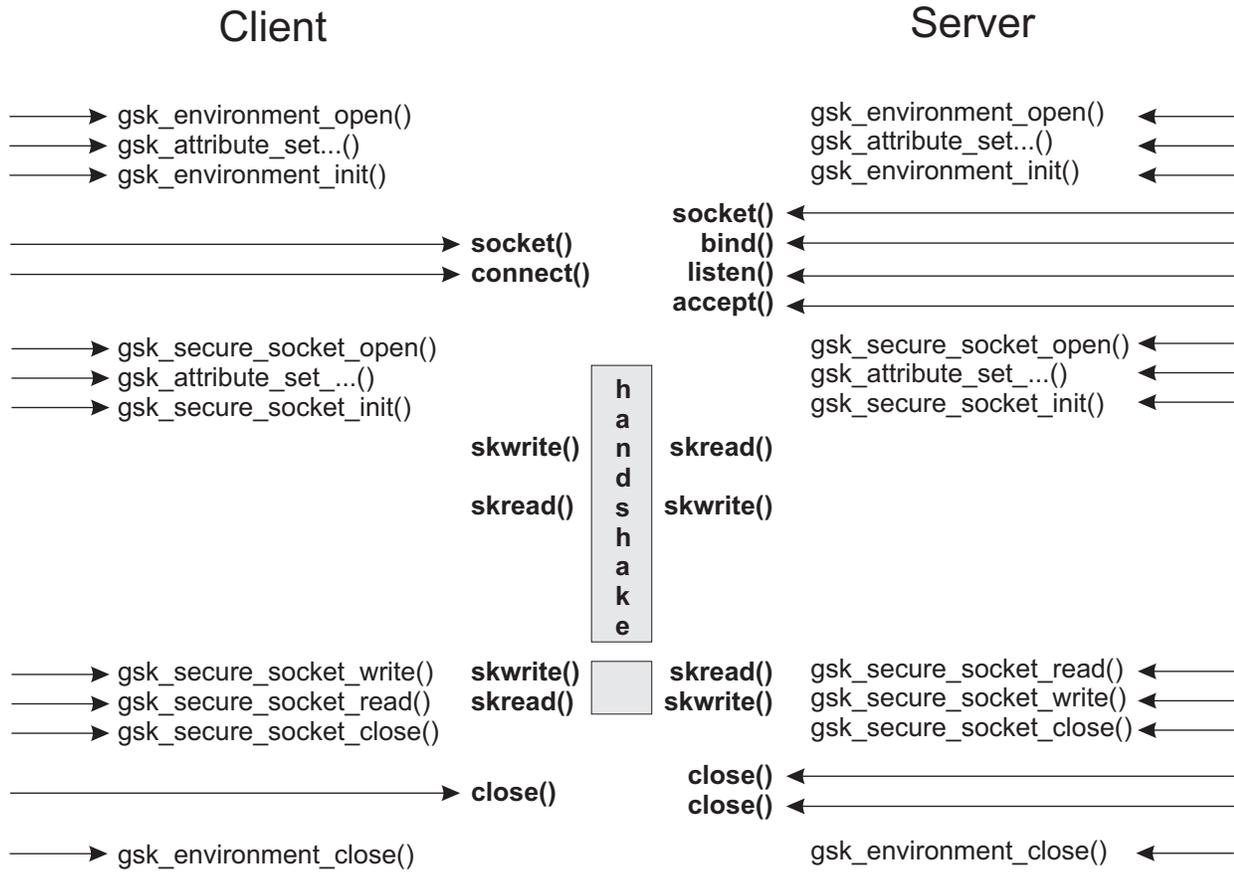


Figure 1. Sockets Programming Model Using System SSL

---

## Chapter 3. Using Hardware Cryptographic Features with System SSL

System SSL uses the Integrated Cryptographic Service Facility (ICSF) if it is available. ICSF provides hardware cryptographic support which will be used instead of the System SSL software algorithms. System SSL checks for the hardware support during its runtime initialization processing and will use the hardware support if it is available. If a severe ICSF error occurs during a cryptographic operation, System SSL will stop using the hardware support and will revert to using the software algorithms.

In order for System SSL to use the hardware support, the ICSF started task must be running and the application userid must be authorized to the following resources in the RACF CSFSERV class, either explicitly or through a generic resource profile:

- CSFCKI - Clear key import
- CSFCKM - Multiple clear key import
- CSFDEC - Symmetric key decrypt
- CSFDSG - Digital signature generate
- CSFDSV - Digital signature verify
- CSFENC - Symmetric key encrypt
- CSFPKB - PKA key build
- CSFPKD - PKA decrypt
- CSFPKE - PKA encrypt
- CSFPKI - PKA key import

Note that access to ICSF cryptographic services can be controlled by the z/OS Security Server (RACF). For further information, refer to the section about controlling who can use cryptographic keys and services in the *z/OS: ICSF Administrator's Guide*, SA22-7521.

For information on the types of hardware cryptographic features supported by ICSF, refer to the *z/OS: ICSF Overview*, SA22-7519. For information on configuring and using ICSF, refer to the *z/OS: ICSF Administrator's Guide* and *z/OS: ICSF System Programmer's Guide*, SA22-7520.

Several IBM products use System SSL. Please check the specific product publications to see if there is information on System SSL and ICSF considerations.



---

## Chapter 4. Writing and Building a z/OS System SSL Application

This chapter describes how to write, build, and run a secure socket layer (SSL) application that uses the System SSL programming interfaces. You can write both client and server applications using the System SSL programming interfaces.

In Release 2 of z/OS, a new set of APIs was introduced, which enhanced the functionality of System SSL. IBM recommends that only the new APIs be used for writing new application programs. Existing application programs should be recoded if possible to use the new APIs. It is also important to note that the new APIs and the deprecated ones are **not** to be mixed in the same application program. For a complete list and descriptions of the new APIs, see Chapter 7, “API Reference” on page 27. Information about the deprecated ones can be found in Chapter 9, “Deprecated Secure Sockets Layer APIs” on page 225.

In addition to writing the SSL applications, you must build a key database for the System SSL source program for a System SSL application to be complete. Refer to Chapter 10, “Certificate/Key Management” on page 249 for details about creating and managing a key database.

Sample programs using the new APIs are shipped in `/usr/lpp/gskssl/examples`.

---

### Writing a System SSL Source Program

The first step in creating a System SSL application is to write the source program using the new System SSL programming interfaces. Refer to Chapter 7, “API Reference” on page 27 for a description of the format of the System SSL programming interfaces.

#### Create an SSL Environment

For both the client and server System SSL programs, you must initialize the System SSL environment using the programming interfaces associated with the SSL environment layer.

##### **gsk\_environment\_open()**

Will define and obtain storage for the SSL environment and return an environment handle to be used on subsequent API invocations.

##### **gsk\_attribute\_set...()**

Sets environment attributes such as:

- The SSL protocol version to be used: SSL Version 2.0, SSL Version 3.0 and/or TLS Version 1.0.
- The key database to be used. (HFS key database file or RACF key ring)
- The password for the key database. This can be specified directly by the application or through the use of a stashed password file. See Chapter 10, “Certificate/Key Management” on page 249 for details about creating a stashed password file.

**Note:** When using RACF key rings, the password and stash file must not be specified.

- The amount of time the SSL session identifier information is valid. By using already negotiated and agreed to SSL session identifier information, System SSL can reduce the amount of data exchanged during the SSL handshake that occurs during the **gsk\_secure\_socket\_init()** call.

##### **gsk\_environment\_init()**

Initializes the SSL environment.

The following example code illustrates how to call the environment layer programming interface from a client or server System SSL program. In this example, SSL Version 3.0 support is requested,

## Writing and Building a z/OS System SSL Application

/keyring/key.kdb is the key database that is used, the password for the key database is "password", and default values are taken for the remaining SSL environment variable attributes.

```
gsk_handle env_handle;
int rc;

/* create the SSL environment */
rc = gsk_environment_open(&env_handle);

/* set environment attributes */
rc = gsk_attribute_set_enum(env_handle, GSK_PROTOCOL_SSLV2, GSK_PROTOCOL_SSLV2_OFF);
/* By default, SSL V2 protocol is set on */
rc = gsk_attribute_set_enum(env_handle, GSK_PROTOCOL_SSLV3, GSK_PROTOCOL_SSLV3_ON);
rc = gsk_attribute_set_enum(env_handle, GSK_PROTOCOL_TLSV1, GSK_PROTOCOL_TLSV1_OFF);
/* By Default, TLS V1.0 protocol is set on */
rc = gsk_attribute_set_buffer(env_handle, GSK_KEYRING_FILE, "/keyring/key.kdb",0);
rc = gsk_attribute_set_buffer(env_handle, GSK_KEYRING_PW, "password",0);

/* initialize environment */
rc = gsk_environment_init(env_handle);
```

The following example code illustrates how to call create an SSL environment for a server System SSL program supporting both SSL Version 2.0, SSL Version 3.0 and TLS Version 1.0.

```
gsk_handle env_handle;
int rc;

/* create the SSL environment */
rc = gsk_environment_open(&env_handle);

/* set environment attributes */
rc = gsk_attribute_set_enum(env_handle, GSK_PROTOCOL_SSLV2, GSK_PROTOCOL_SSLV2_ON);
rc = gsk_attribute_set_enum(env_handle, GSK_PROTOCOL_SSLV3, GSK_PROTOCOL_SSLV3_ON);
rc = gsk_attribute_set_enum(env_handle, GSK_PROTOCOL_TLSV1, GSK_PROTOCOL_TLSV1_ON);
rc = gsk_attribute_set_buffer(env_handle, GSK_KEYRING_FILE, "/keyring/key.kdb",0);
rc = gsk_attribute_set_buffer(env_handle, GSK_KEYRING_PW, "password",0);

/* initialize environment */
rc = gsk_environment_init(env_handle);
```

**Note:** Once the environment is initialized, the environment attributes cannot be changed unless they are also attributes of the secure socket connection. In this case, they can be changed only for that connection. If changes are necessary to the environment, a new SSL environment can be created within the same process.

Once the System SSL program has successfully created the SSL environment, it must now perform the steps needed to allow the program to communicate with a peer program. The exact sockets and System SSL calls required to allow the program to communicate differ depending on whether the program is a client or a server.

## System SSL Server Program

You can use the following sockets and System SSL calls to enable a server program to communicate with a client program.

To create a stream socket to which client programs can connect, use the following function call:

```
int server_sock;

server_sock = socket(AF_INET, SOCK_STREAM, 0);
```

Now that the server program socket has been created, bind the socket to a port (for example, 1234) that is known to the client program using the following function call:

## Writing and Building a z/OS System SSL Application

```
int rc;
int namelength;
struct sockaddr_in name;

nameLength = sizeof(name);
memset(&name, '\0', nameLength);
name.sin_family = AF_INET;
name.sin_port = 1234;
name.sin_addr.s_addr = INADDR_ANY;

rc = bind(server_sock, (struct sockaddr *)&name, nameLength);
```

To make the server program socket ready to listen for incoming connection request, use the following function call:

```
int rc;

rc = listen(server_sock, 5); /* allow max of 5 connections */
```

The server program is now ready to begin accepting connections from client programs. To accept connections, use the following function calls:

```
int client_sock;
int incomingNameLength;
struct sockaddr_in incomingName;

client_sock = accept(server_sock, (struct sockaddr *)&incomingName, &incomingNameLength);
```

After successfully accepting a connection from a client program, the server program must establish the secure socket connection which will result in the SSL handshake being performed. Once the handshake is completed, secure transfer of application data can be done. The secure socket connection will be established with the following attribute values:

- The socket descriptor over which the communication is to occur.
- Certificate with label "ServerCertLabel"
- The type of handshake (for example, server) to be performed.
- The set of SSL protocol cipher specs to be allowed for the secure session. The cipher is selected by the System SSL Server program according to the System SSL Server's order of usage preference.
- The address of a routine to be called by System SSL to read data from the socket for the secure session.
- The address of a routine to be called by System SSL to write data on the socket for the secure session.

```
gsk_handle soc_handle;
int rc;
gsk_iocallback local_io = {secureSocRecv, secureSocSend, NULL, NULL, NULL, NULL};

rc = gsk_secure_socket_open(env_handle, &soc_handle);

rc = gsk_attribute_set_numeric_value(soc_handle, GSK_FD, client_sock);
rc = gsk_attribute_set_buffer(soc_handle, GSK_KEYRING_LABEL, "ServerCertLabel",0);
rc = gsk_attribute_set_enum(soc_handle, GSK_SESSION_TYPE, GSK_SERVER_SESSION);
rc = gsk_attribute_set_buffer(soc_handle, GSK_V2_CIPHER_SPECS, "6321",0);
rc = gsk_attribute_set_buffer(soc_handle, GSK_V3_CIPHER_SPECS, "0906030201",0);
rc = gsk_attribute_set_callback(soc_handle, GSK_IO_CALLBACK, &local_io);

rc = gsk_secure_socket_init(soc_handle);
```

The System SSL program should provide the function to send and receive data over the application socket. For more information, see "I/O Routine Replacement" on page 19. Use the following function calls, **send** and **recv**, to send and receive the application data.

```
int secureSocRecv(int fd, void *data, int len, char *user_data) {
    return( recv( fd, data, len,0 ));
}
```

## Writing and Building a z/OS System SSL Application

```
int secureSocSend(int fd, void *data, int len, char *user_data) {
    return(send( fd, data, len,0 ));
}
```

After the server program has successfully called **gsk\_secure\_socket\_init()**, it can now read and write data securely over the application socket. To read application data from the application socket, use the following code:

```
int rc;
int buffer_length;
int length_read;
char *data_buffer;

rc = gsk_secure_socket_read(soc_handle, data_buffer, buffer_length, &length_read);
```

To write application data over the application socket, use the following code:

```
int rc;
int buffer_length;
int length_written;
char *data_buffer;

rc = gsk_secure_socket_write(soc_handle, data_buffer, buffer_length, &length_written);
```

Once the server program is finished using the application socket to securely send and receive data, it must free all of the System SSL resources for the SSL session and close the socket. To free the System SSL resource for the SSL session, use the following **gsk\_secure\_socket\_close()** call:

```
gsk_secure_socket_close(&soc_handle);
```

To free the resources used by the SSL environment, use the following **gsk\_environment\_close()** call:

```
gsk_environment_close(&env_handle);
```

Finally, to close the application socket, use the following function call:

```
int rc;
rc = close(client_sock);
```

## System SSL Client Program

The sockets and System SSL calls a client program uses are very similar to the calls the server program uses. Rather than accepting connections like a server program, a client program connects to the server program.

To create a stream socket that the client program can use to connect to the server, use the following function call:

```
int sock;

sock = socket(AF_INET, SOCK_STREAM,0);
```

Now that the client program socket has been created, connect the socket to the server program port using the following function call:

```
int rc;
int namelength;
struct sockaddr_in name;
char *ServeHostName;

nameLength = sizeof(name);
memset(&name, '\\0', nameLength);
name.sin_family = AF_INET;
name.sin_port = 1234;
name.sin_addr.s_addr = ServerHostName;
rc = connect(sock, (struct sockaddr *)&name, nameLength);
```

## Writing and Building a z/OS System SSL Application

After successfully connecting to the server program, the client program must establish the secure socket connection. This connection will cause the SSL handshake to be performed. Once the handshake is complete, secure communication of the application data can be done. The following example code establishes the connection using the following attribute values:

- The socket descriptor over which the communication is to occur.
- Certificate with label "THELABEL"
- The type of handshake (client) to be performed.
- The set of SSL protocol cipher specs to be allowed for the secure session in client-preferred order.

**Note:** Although the client is allowed to specify a preference order, an SSL server may or may not honor the preference.

- The address of a routine to be called by System SSL to read data from the socket for the secure session.
- The address of a routine to be called by System SSL to write data on the socket for the secure session.

```
int rc;
gsk_handle soc_handle;
gsk_iocallback local_io = {secureSocRecv, secureSocSend, NULL, NULL, NULL, NULL};

rc = gsk_secure_socket_open(env_handle, &soc_handle);
rc = gsk_attribute_set_numeric_value(soc_handle, GSK_FD, client_sock);
rc = gsk_attribute_set_buffer(soc_handle, GSK_KEYRING_LABEL, "THELABEL",0);
rc = gsk_attribute_set_enum(soc_handle, GSK_SESSION_TYPE, GSK_CLIENT_SESSION);
rc = gsk_attribute_set_buffer(soc_handle, GSK_V2_CIPHER_SPECS, "6321",0);
rc = gsk_attribute_set_buffer(soc_handle, GSK_V3_CIPHER_SPECS, "0906030201",0);
rc = gsk_attribute_set_callback(soc_handle, GSK_IO_CALLBACK, &local_io);

rc = gsk_secure_socket_init(soc_handle);
```

The System SSL program should provide the function to send and receive data over the application socket. For more information, see "I/O Routine Replacement" on page 19. Use the following function calls, **send** and **recv**, to send and receive the application data.

```
int secureSocRecv(int fd, void *data, int len, char *user_data) {
    return( recv( fd, data, len,0 ));
}

int secureSocSend(int fd, void *data, int len, char *user_data) {
    return(send( fd, data, len,0 ));
}
```

After the client program has successfully called **gsk\_secure\_socket\_init()**, it can now read and write data securely over the application socket. To read application data from the application socket, use the following code:

```
int rc;
int buffer_length;
int length_read;
char *data_buffer;

rc = gsk_secure_socket_read(soc_handle, data_buffer, buffer_length, &length_read);
```

To write application data over the application socket, use the following code:

```
int rc;
int buffer_length;
int length_written;
char *data_buffer;

rc = gsk_secure_socket_write(soc_handle, data_buffer, buffer_length, &length_written);
```

Once the client program is finished using the application socket to securely send and receive data, it must free all of the System SSL resources for the SSL session and close the socket.

## Writing and Building a z/OS System SSL Application

To free the System SSL resource for the SSL session, use the following `gsk_secure_socket_close()` call:

```
gsk_secure_socket_close(&soc_handle);
```

To free the resources used by the SSL environment, use the following `gsk_environment_close()` call:

```
gsk_environment_close(&env_handle);
```

Finally, to close the application socket, use the following function call:

```
int rc;  
rc=close(sock);
```

---

## Building a z/OS System SSL Application

1. Write the System SSL source program (see “Writing a System SSL Source Program” on page 11).
2. Compile your System SSL source program.
3. Specify `/usr/lib/GSKSSL.x` in link-edit step.
4. Build a key database file using the `gskkyman` utility or create a RACF key ring using the `RACDCERT` command. The name of the key database file or RACF key ring must match the name you specified as the `GSK_KEYRING_FILE` on the `gsk_attribute_set_buffer()` API. You need the name of the key database file or RACF key ring and for key database files either the password associated with the key file or the stash file name. The password must match the password specified on `GSK_KEYRING_PW` on the `gsk_attribute_set_buffer()` API or must be set to `NULL` if using a RACF key ring. Refer to Chapter 10, “Certificate/Key Management” on page 249 for information on how to create a key database file or RACF key ring.

---

## Running a z/OS System SSL Application

After successfully writing and building the System SSL application and creating the key database, you can run the System SSL application. To run the application follow these steps:

1. If not in the linklist or LPA, ensure that the STEPLIB identifies the PDS name (*pdsname*.SGSKLOAD) that contains the System DLLs. For example, in the OS/390 shell, issue the following command:

```
export STEPLIB=$STEPLIB:pdsname.SGSKLOAD
```

2. Ensure that the key database file or RACF key ring is accessible to the System SSL application.
3. Run the System SSL application.

---

## Additional Topics

### Non-Blocking I/O

Applications wishing to communicate securely to one another may establish a secure connection. Each application opens a socket and attempts to establish an SSL connection. After an SSL connection has been established, the applications may now use the socket to exchange data securely. The default (blocking) mode of a socket requires an application attempting to read or write to the socket to block until all expected data has been received. This blocking may not be desirable since no other processing may occur while the application is waiting for a read or write to complete. One solution to this problem is the use of non-blocking sockets.

When a socket is setup as non-blocking, reads and writes to the socket do not cause the application to block and wait. Instead the read or write function will read/write only the data currently available (if any). If the entire read/write is not completed, a status indicator is returned. The application may retry the read/write later.

## Non-Blocking Socket Primer

When a server wishes to communicate with clients via a socket, the following routines are used:

Table 2. Server Communicating with Clients Via a Socket

Routine	Purpose
1) socket()	Create a socket
2) bind()	Register the socket
3) listen()	Indicate willingness to accept connections
4) accept()	Accept a connection request
5) Read request	
6) Write response	
7) Return to step 4	

Once the **accept()** routine is called, the server will block until data is available for the socket. Problems arise when the server wishes to monitor multiple sockets simultaneously or if the server wishes to perform other tasks until data is available on the socket. However, by configuring the socket as non-blocking, these problems may be avoided. For more information, see “Enable/Disable Non-Blocking Mode” on page 18. When using non-blocking sockets, the **select()** routine is used to instruct the system to notify the server application when data is available on a particular socket.

Table 3. Using the select() Routine

Routine	Purpose
1) socket()	Create a socket
2) bind()	Register the socket
3) listen()	Indicate willingness to accept connections
4) Set socket as non-blocking	See “Enable/Disable Non-Blocking Mode” on page 18
5) select()	Monitor a number of sockets
6) accept()	Accept a connection request
7) Read request	If unable to read all data, return to step 5
8) Write response	If unable to write all data, return to step 5
9) Return to step 4	

## Affected SSL Functions

The following functions are affected by the use of non-blocking sockets with SSL.

### gsk\_secure\_socket\_init()

During the SSL handshake, the **io\_setsocketoptions()** routine is called by the **gsk\_secure\_socket\_init()** routine before initiating the SSL handshake (GSK\_SET\_SOCKET\_STATE\_FOR\_HANDSHAKE) and again upon completion of the SSL handshake (GSK\_SET\_SOCKET\_STATE\_FOR\_READ\_WRITE). The default **io\_setsocketoptions()** routine puts the socket into blocking mode for GSK\_SET\_SOCKET\_STATE\_FOR\_HANDSHAKE and restores the original mode for GSK\_SET\_SOCKET\_STATE\_FOR\_READ\_WRITE. In order to perform a non-blocking SSL handshake, an application supplied **io\_setsocketoptions()** callback must be provided to control the state of the socket. When the socket is in non-blocking mode, **gsk\_secure\_socket\_init()** may return GSK\_WOULD\_BLOCK\_READ or GSK\_WOULD\_BLOCK\_WRITE. This error indicates that System SSL was unable to read or write the entire message. When this occurs, the application should call **select()** and then call **gsk\_secure\_socket\_init()** again.

## Writing and Building a z/OS System SSL Application

`gsk_secure_socket_read()`

Once the socket has been configured as non-blocking, any calls to **`gsk_secure_socket_read()`** can potentially return `GSK_WOULD_BLOCK`. When this occurs, the application should call **`select()`** and then call **`gsk_secure_socket_read()`** again.

`gsk_secure_socket_write()`

Once the socket has been configured as non-blocking, any calls to **`gsk_secure_socket_write()`** can potentially return `GSK_WOULD_BLOCK`. When this occurs, the application should call **`select()`** and then call **`gsk_secure_socket_write()`** again.

**Enable/Disable Non-Blocking Mode:** Once a socket has been created using the **`socket()`** call, it may be set to non-blocking as follows:

```
#include "sys/ioctl.h"
int on =1;
int off =0;

//Enable non-blocking
ioctl (mySocket, FIONBIO, &(on));
//Disable non-blocking
ioctl (mySocket, FIONBIO, (char *) &(off));
```

### Differences in SSL and Unsecured Non-Blocking Mode:

#### Partial Data

An unsecured socket in non-blocking mode will return the partial data received or written. Since System SSL processes encrypted data, it is not possible to decrypt a message until the entire message has been received, making it impossible to return partial data.

#### Error Indicator

When non-blocking mode is used on a non-secure socket, the status indicator is generally found by checking the *errno* variable, which is normally `EWOULDBLOCK`. System SSL does **not** set the *errno* variable. Instead the value returned from **`gsk_secure_socket_read()`** or **`gsk_secure_socket_write()`** is set to `GSK_WOULD_BLOCK`. **`gsk_secure_socket_init()`** will return either `GSK_WOULD_BLOCK_READ` or `GSK_WOULD_BLOCK_WRITE`.

## Client Authentication Certificate Selection

SSL enables the application to prompt the client user to select a certificate from a list during the client authentication process in the SSL handshake.

This is accomplished with a registered callback routine that is invoked from inside the **`gsk_secure_socket_init()`** function call. This section provides an overview of that code.

The client application code must provide the following functions:

- Register a standard C linkage callback routine using the **`gsk_attribute_set_callback()`** function call.
- Implement the callback routine that performs these functions:
  - Get the list of available certificates using the **`gsk_attribute_get_data()`** function call with the `GSK_DATA_ID_SUPPORTED_KEYS` option. This returns a list of labels from the key data base file or RACF key ring.
  - Display the list of labels to the user.
  - Prompt the user to select the label from the list
  - Set the label to be used with a **`gsk_attribute_set_buffer()`** function call with the `GSK_KEYRING_LABEL` option.
  - Return to SSL with the return value set to indicate use client authentication.
  - If the user elects to not use any of the certificates in the list, return with the value set to skip client authentication. A certificate will not be sent to the partner, but the SSL handshake will complete. The server will decide whether to continue or close the connection.

- Optionally, the application can display certificate information using the **gsk\_get\_cert\_by\_label()** function call.
- Optionally, the application can use the **gsk\_attribute\_get\_data()** function call with the **GSK\_DATA\_ID\_SERVER\_ISSUERS** option to display a list of server signer certificates.

## I/O Routine Replacement

### Callback Routine for IO

SSL allows applications to specify how I/O is to take place. This is done by specifying callback routines for receiving and sending data. The contents of this routine can be very unique per application. SSL has an internally defined default routine which is used if **gsk\_attribute\_set\_callback()** is not used to override I/O routines. The default assumes that TCP/IP is being used. For reading it will execute a **recv()** and for write a **send()**. If not using TCP/IP, applications should also consider the specification of the **getpeername** and **setsocketoptions** callback routine. It also depends on TCP/IP as being the transport layer protocol.

**Note:** Application provided I/O routines must use standard C linkage conventions.

### Use of User Data

Some complex applications require application-specific data to be available in the SSL callbacks. SSL enables this with the **gsk\_attribute\_set\_buffer()** and **gsk\_attribute\_get\_buffer()** function calls. In addition, the I/O callbacks pass a pointer to the user data.

The following are the steps that need to be taken to effectively use the user data functions:

- Issue the **gsk\_secure\_socket\_open()** function. This will return a *soc\_handle*.
- To set the user data for a connection issue:
  - **gsk\_attribute\_set\_buffer(*soc\_handle*, **GSK\_USER\_DATA**, *user\_data*, *sizeof(user\_data)*);**
  - This function call copies the *user\_data* into an area of storage owned by SSL.
- The address of the SSL copy of the user data is passed as a parameter to the user-specified **read**, **write**, **getpeername**, and **set\_socket\_options** callbacks.
- Other callbacks pass the *soc\_handle* as a parameter to the callback. To find the address of the copy of user data associated with a particular connection, issue:
  - **gsk\_attribute\_get\_buffer(*soc\_handle*, **GSK\_USER\_DATA**, *&user\_data\_ptr*, *&user\_data\_size*);**
  - You can modify the contents of the SSL copy of the user data, but you may not free or re-allocate the SSL user data. The SSL user data will be freed when the connection is closed with the **gsk\_secure\_socket\_close()** function call.

You can point to other application data from the SSL user data area. However, it is up to the application to free this other application data before the connection is closed.

### Session ID (SID) Cache Replacement

The SSL protocol has a mechanism built in to allow for faster secure connections between a client/server pair. There is a concept of a SSL Session that allows this to happen. The first time a client and server connect, cryptographic characteristics of that connection are saved into a Session Cache buffer. A Session is identified by a Session ID (SID). The cached cryptographic components (SID cache entry) allows for new bulk encryption keys to be generated with subsequent SSL handshakes between the same client/server pair. The subsequent handshakes would be abbreviated since much of the data used to generate keys is in the SID cache entry. This abbreviated handshake does not require public key encryption to take place.

Public key encryption is very time consuming thus avoiding it is a great performance boost for servers using SSL. A SID Cache entry exists for a limited time. Care should be used when specifying how long a SSL session is allowed to live. Setting the SID cache timeout or number of SID cache buffer entries to ZERO will turn off SID caching causing a full handshake to be completed for every connection.

## Writing and Building a z/OS System SSL Application

Applications need to be sensitive to both security and performance issues. Security conscious applications should keep the session timeout values very low to ensure keys are generated frequently to avoid security breaches. Applications that are more performance conscious than security conscious should have longer session timeouts and a larger cache size.

### Session ID (SID)

Modifying SSL session caching parameters can be used to help tune the security performance characteristics of SSL enabled servers. SSL internally does session caching and is controlled only by setting the length of a SSL session and the number of entries in the buffer. The internal SSL SID cache is fixed to a configurable number of entries. There is no way to remove or to re-use entries for other connections except for repeated connections between the same client/server pair. The list of options for extending SID caching functionality can become quite long so an external SID cache buffer API was created for those who are more discriminating about managing SID cache data. There are several callbacks used for external SID cache buffer access.

It should be noted that there are probably few applications where using an external SID cache makes sense. Some recommended environments where it might be considered is in a server configuration where multiple instances of a server exist for work load balancing purposes. It might be desirable to have a single SID cache buffer to be used by all of the processes which each server is running in. Usually this can be avoided by writing applications which are multi threaded. All threads would use the single internal SID cache buffer.

#### Format:

```
typedef gsk_data_buffer *      (*ptgsk_getcache) (
    const unsigned char *      session_id,
    unsigned int                session_id_length,
    int                         ssl_version);

typedef gsk_data_buffer *      (*ptgsk_putcache) (
    gsk_data_buffer *          ssl_session_data,
    const unsigned char *      session_id,
    unsigned int                session_id_length,
    int                         ssl_version);

typedef void                   (*ptgsk_deletcache) (
    const unsigned char *      session_id,
    unsigned int                session_id_length,
    int                         ssl_version);

typedef void                   (*ptgsk_freecache) (
    gsk_data_buffer *          ssl_session_data);

typedef struct _gsk_sidcache_callback {
    ptgsk_getcache              Get;
    ptgsk_putcache              Put;
    ptgsk_deletcache            Delete;
    ptgsk_freecache             FreeDataBuffer;
} gsk_sidcache_callback;
```

#### Callbacks:

##### Get

Specifies the routine System SSL calls to search the session ID cache for the entry that matches the passed values in **sessionID**, **sessionIDLen**, and **SSLVersion**. The value returned by this routine is a pointer to a malloc'ed **gsk\_data\_buffer** structure for the **sslSessionData** that contains the session id cache entry.

##### Put

Specifies the routine System SSL calls to add an entry to the session ID cache. The passed in values **sessionID**, **sessionIDLen**, **SSLVersion** and **sslSessionData** are used to define the entry. This

## Writing and Building a z/OS System SSL Application

routine is responsible for getting storage to hold the entry. The value returned by this routine is either NULL if unable to allocate storage or a pointer to a **gsk\_data\_buffer** structure containing the **sslSessionData** that was passed into the routine.

### Delete

Specifies the routine System SSL calls to delete an entry from the session ID cache. **sessionID**, **sessionIDL** and **SSLVersion** are used to determine which entry is deleted.

### FreeDataBuffer

Specifies the routine that System SSL calls to free memory that was returned by the **Get** session id cache callback routine.

### Parameters:

#### **sessionID**

The buffer containing the Session data

#### **sessionIDL**

The length of the entry for the SID cache buffer entry.

#### **SSLVersion**

The version of the SSL Protocol.

#### **data**

This is the buffer that is created by the external SID cache process to transfer the SID cache entry to SSL.

## Writing and Building a z/OS System SSL Application

---

## Chapter 5. Migrating to the New SSL Interfaces

In Release 2 of z/OS, a new set of APIs were added to provide enhanced function and flexibility, and supersede those APIs introduced in previous System SSL releases. IBM recommends that only the new APIs be used for writing new application programs. Existing application programs should be recoded if possible to use the new APIs. This chapter describes how to migrate your existing application programs to use the new SSL interfaces.

**Note:** When migrating to use any of the new APIs, the entire System SSL application must be migrated to use only the new APIs. The application must not contain a mixture of deprecated and new APIs.

If you are migrating from z/OS System SSL V1R1 or earlier, the sample program shipped with this release is the same program that was shipped with the previous release, with all API usage migrated to the new APIs.

- Replace manually initializing the *gsk\_init\_data* structure with **gsk\_environment\_open()**, plus a number of **gsk\_attribute\_set\_buffer()**, **gsk\_attribute\_set\_enum()** and **gsk\_attribute\_set\_numeric\_value()** functions (as needed) to set attributes.
- Replace **gsk\_get\_cipher\_info()** with a call to **gsk\_attribute\_get\_buffer()** to get the list of available ciphers. This call must be done after a successful **gsk\_environment\_open()** call. The ciphers returned always represent the high security ciphers.
- Replace **gsk\_initialize()** with **gsk\_environment\_init()**.
- Replace manually initializing the *gsk\_soc\_init\_data* structure with **gsk\_secure\_socket\_open()**, plus a number of **gsk\_attribute\_set\_buffer()**, **gsk\_attribute\_set\_enum()** and **gsk\_attribute\_set\_numeric\_value()** functions (as needed) to set attributes.
- Replace manually initializing the *gsk\_soc\_init\_data* structure with the addresses of your I/O callback routines with **gsk\_attribute\_set\_callback()**. You specify the address of a *gsk\_iocallback* structure that contains the addresses of the callback routines. The *gsk\_iocallback* structure is defined in **gskssl.h**. Note that an additional parameter must be added to the function declarator for your existing callback routines.
- Replace **gsk\_user\_set()** with **gsk\_attribute\_set\_callback()** for defining the address of your get peer ID callback routine. You specify the address of an *gsk\_iocallback* structure that contains the address of the callback routine. The *gsk\_iocallback* structure is defined in **gskssl.h**. Note that an additional parameter must be added to the function declarator for your existing callback routine.
- Replace **gsk\_user\_set()** with **gsk\_attribute\_set\_callback()** for defining the address of your session ID cache callback routines. You specify the address of a *gsk\_sidcache\_callback* structure that contains the address of the callback routines. The *gsk\_sidcache\_callback* structure is defined in **gskssl.h**.
- Replace **gsk\_get\_dn\_by\_label()** with **gsk\_get\_cert\_by\_label()**.
- Replace **gsk\_secure\_soc\_init()** with **gsk\_secure\_socket\_init()**.
- Replace **gsk\_secure\_soc\_read()** with **gsk\_secure\_socket\_read()**. Note that **gsk\_secure\_socket\_read()** has an extra parameter to return the length of the data read.
- Replace **gsk\_secure\_soc\_write()** with **gsk\_secure\_socket\_write()**. Note that **gsk\_secure\_socket\_write()** has an extra parameter to return the length of the data written.
- Replace **gsk\_secure\_soc\_close()** with **gsk\_secure\_socket\_close()**.
- Be sure that every **gsk\_secure\_socket\_open()** is matched with a **gsk\_secure\_socket\_close()** even if there is an error on **gsk\_environment\_init()**. Normal sequence is **open, init, close**. So, if **init** gets an error return code, you still must do the **close**.
- Be sure that every **gsk\_environment\_open()** is matched with a **gsk\_environment\_close()** even if there is an error on **gsk\_secure\_socket\_init()**. Normal sequence is **open, init, close**. So, if **init** gets an error return code, you still must do the **close**.
- A method is provided to display certificates after **gsk\_secure\_socket\_init()** has been issued. You may use **gsk\_attribute\_get\_cert\_info()** if desired.

- Be aware that all of the error return values have been renamed and renumbered. Program logic will have to be changed accordingly.
- There is a new **gsk\_strerror()** debug routine that returns a text string (in English only) when an error number is passed to it.

---

## Chapter 6. Building and Running a Java System SSL Application

This chapter describes how to write, build, and run a Java application that uses Java Socket calls and the Java System SSL classes.

### Important Note

For z/OS V1R4, Java support is being deprecated. For Java SSL support, please use IBM JSSE.

---

### Writing a Java Source Program

Java applications that wish to take advantage of System SSL need to use the shipped GSKSSL.jar file contained in /usr/lpp/gskssl/java. This file provides the front end to all the System SSL functionality. These applications need to be written and executed using Java 1.3.1 or higher. Online documentation is available in HTML format in directory /usr/lpp/gskssl/java/docs. This documentation will aid in finding information about the various methods and properties of the classes contained in the GSKSSL.jar file.

Following are the steps for creating a Java program that uses System SSL:

1. Create a new SSLSession object:

```
SSLSession session = new SSLSession();
```

2. Create a normal Java Socket:

```
Socket soc = new Socket(host,port);
```

3. Manipulate the various session properties, for example:

```
session.setSecType("SSLV3");  
session.setKeyring("key1.kdb");  
session.setKeyringPw("password");
```

4. Perform the equivalent to gsk\_initialize(), which uses all the properties that are set in the SSLSession class:

```
session.initialize();
```

5. Perform the equivalent to gsk\_secure\_soc\_init(), which uses all the properties that are set in the SSLSession class:

```
session.init(soc);
```

6. Obtain the input and output streams to the SSL session:

```
SSLInputStream in = new SSLInputStream(session,soc.getInputStream());  
SSLOutputStream out = new SSLOutputStream(session);
```

7. Read and write to the respective SSLOutputStream and SSLInputStream. This is done similarly to how you would read and write to a normal Java InputStream or OutputStream:

```
//example of how to write data to the SSLOutputStream  
String outString;  
outString=new String( "running" );  
byte [] outBuf = outString.getBytes();  
out.write(outBuf.length);  
out.write(outBuf, 0, outBuf.length);  
out.flush();
```

```
//example of how to read from the SSLInputStream  
int msgLen = in.read();  
byte [] inBuf = new byte[msgLen];  
in.read(inBuf, 0, msgLen);
```

8. Once the data is transferred, all that is left to do is close the SSLSession object and the Java socket:

```
session.close();  
soc.close();
```

## Building and Running a Java System SSL Application

---

### Building a z/OS Java System SSL Application

1. Write the Java source program (see “Writing a Java Source Program” on page 25).
  2. Compile your Java source program with the GSKSSL.jar file included in the classpath.
- 

### Running a z/OS Java System SSL Application

1. Complete the steps in “Building a z/OS Java System SSL Application”.
2. Set the LIBPATH environment variable to contain the path to the libSSLSocketLib.so file, which is in the /usr/lpp/gskssl/lib directory.
3. Run the Java application with the GSKSSL.jar file in the classpath.

---

## Chapter 7. API Reference

This chapter describes the set of application programming interfaces (APIs) that z/OS System SSL supports for performing secure sockets layer (SSL/TLS) communication.

These APIs were introduced in z/OS Version 1 Release 2 and beyond and supersede the APIs from prior releases. **Only** the APIs in this chapter should be used for writing new application programs. Existing application programs should be recoded if possible to use the new APIs. See Chapter 5, “Migrating to the New SSL Interfaces” on page 23 for more information about updating your application programs.

The deprecated APIs included in Chapter 9, “Deprecated Secure Sockets Layer APIs” on page 225 are for **reference only**. When creating new application programs, you **must not** include any of the deprecated APIs; you should use **only** the APIs in this chapter.

The following documents provide more information on X.509 certificates and the Secure Sockets Layer protocol:

- FIPS 186 - Digital Signature Standard
- PKCS #1 - RSA Encryption Standard
- PKCS #5 - Password-based Encryption
- PKCS #7 - Cryptographic Message Syntax
- PKCS #8 - Private Key Information Syntax
- PKCS #10 - Certification Request
- PKCS #12 - Personal Information Exchange
- RFC 2246 - Transport Layer Security (TLS) Version 1
- RFC 2253 - String Representation of Distinguished Names
- RFC 2279 - UTF-8 (UCS Transformation Format 8)
- RFC 2459 - X.509 certificate, certificate revocation list, and certificate extensions
- RFC 2587 - PKIX LDAP Version 2 Schema

Following is a list of APIs. Use these APIs when creating new application programs. If possible, recode your existing application programs to use these APIs as well:

- **gsk\_attribute\_get\_buffer()** (see page 29 )
- **gsk\_attribute\_get\_cert\_info()** (see page 31)
- **gsk\_attribute\_get\_data()** (see page 34)
- **gsk\_attribute\_get\_enum()** (see page 36)
- **gsk\_attribute\_get\_numeric\_value()** (see page 38)
- **gsk\_attribute\_set\_buffer()** (see page 40)
- **gsk\_attribute\_set\_callback()** (see page 43 )
- **gsk\_attribute\_set\_enum()** (see page 47)
- **gsk\_attribute\_set\_numeric\_value()** (see page 49)
- **gsk\_environment\_close()** (see page 51)
- **gsk\_environment\_init()** (see page 52)
- **gsk\_environment\_open()** (see page 54)
- **gsk\_free\_cert\_data()** (see page 58)
- **gsk\_get\_cert\_by\_label()** (see page 59)
- **gsk\_get\_cipher\_suites()** (see page 62)
- **gsk\_get\_update()** (see page 63)

- | • **gsk\_list\_free()** (see page 64)
- | • **gsk\_secure\_socket\_close()** (see page 65)
- | • **gsk\_secure\_socket\_init()** (see page 66 )
- | • **gsk\_secure\_socket\_misc()** (see page 69)
- | • **gsk\_secure\_socket\_open()** (see page 71)
- | • **gsk\_secure\_socket\_read()** (see page 72)
- | • **gsk\_secure\_socket\_shutdown()** (see page 74)
- | • **gsk\_secure\_socket\_write()** (see page 76)
- | • **gsk\_strerror()** (see page 78 )

## gsk\_attribute\_get\_buffer()

Gets the value of an attribute buffer.

### Format

```
#include <gskssl.h>

gsk_status gsk_attribute_get_buffer (
    gsk_handle      ssl_handle,
    GSK_BUF_ID     buffer_id,
    const char **  buffer_value,
    int *          buffer_length)
```

### Parameters

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*buffer\_id*

Specifies the buffer identifier.

*buffer\_value*

The buffer is in storage owned by the SSL runtime and must not be modified or released by the application. The buffer returned for the GSK\_USER\_DATA identifier may be modified by the application but must not be released.

*buffer\_length*

Returns the length of the buffer value.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ATTRIBUTE\_INVALID\_ID]

The buffer identifier is not valid or cannot be used with the specified handle.

#### [GSK\_INVALID\_HANDLE]

The handle is not valid.

#### [GSK\_INVALID\_STATE]

The handle is closed.

### Usage

The **gsk\_attribute\_get\_buffer()** routine will return a buffer value for an SSL environment or an SSL connection. The buffer is in storage owned by the SSL runtime and must not be released by the application. The address will remain valid until the SSL environment or connection is closed or until the application calls the **gsk\_attribute\_set\_buffer()** routine to set a new buffer value.

The following buffer identifiers are supported:

#### GSK\_CONNECT\_CIPHER\_SPEC

Returns the cipher specification selected for an initialized connection. Refer to the description of the **gsk\_environment\_open()** routine for a list of valid cipher specifications.

GSK\_CONNECT\_CIPHER\_SPEC may be specified only for an SSL connection.

#### GSK\_CONNECT\_SEC\_TYPE

Returns the security protocol for an initialized connection. The value will be "SSLV2", "SSLV3", or

### **gsk\_attribute\_get\_buffer()**

"TLSV1" depending upon the protocol selected during the SSL handshake.  
GSK\_CONNECT\_SEC\_TYPE may be specified only for an SSL connection.

### **GSK\_KEYRING\_FILE**

Returns the name of the key database HFS file or the SAF key ring. A key database is used if a database password or stash file is defined using either an environment variable or the **gsk\_attribute\_set\_buffer()** routine.

### **GSK\_KEYRING\_LABEL**

Returns the label associated with the certificate being used by the SSL environment or connection. This will be the value set by the application if the environment or connection is not initialized. GSK\_KEYRING\_LABEL may be specified for an SSL environment or an SSL connection.

### **GSK\_KEYRING\_PW**

Returns the password for the key database. A NULL address will be returned after the environment is initialized. GSK\_KEYRING\_PW may be specified only for an SSL environment.

### **GSK\_KEYRING\_STASH\_FILE**

Returns the name of the key database password stash file. GSK\_KEYRING\_STASH\_FILE may be specified only for an SSL environment.

### **GSK\_LDAP\_SERVER**

Returns the DNS name or IP address of the LDAP server. GSK\_LDAP\_SERVER may be specified only for an SSL environment.

### **GSK\_LDAP\_USER**

Returns the distinguished name to use when connecting to the LDAP server. GSK\_LDAP\_USER may be specified only for an SSL environment.

### **GSK\_LDAP\_USER\_PW**

Returns the password to use when connecting to the LDAP server. GSK\_LDAP\_USER\_PW may be specified only for an SSL environment.

### **GSK\_SID\_VALUE**

Returns the session identifier for an initialized connection. This is the Base64-encoded version of the session identifier and consists of displayable characters. GSK\_SID\_VALUE may be specified only for an SSL connection.

### **GSK\_USER\_DATA**

Returns the address of the user data to be passed to SSL exit routines. The application may alter the user data but may not free it. GSK\_USER\_DATA may be specified only for an SSL connection.

### **GSK\_V2\_CIPHER\_SPECS**

Returns the SSL V2 cipher specifications as a string consisting of 1-character values. GSK\_V2\_CIPHER\_SPECS may be specified for an SSL environment or an SSL connection. Refer to the description of **gsk\_environment\_open()** for a list of valid cipher specifications.

### **GSK\_V3\_CIPHER\_SPECS**

Returns the SSL V3 cipher specifications as a string consisting of 2-character values. GSK\_V3\_CIPHER\_SPECS may be specified for an SSL environment or an SSL connection. Refer to the description of **gsk\_environment\_open()** for a list of valid cipher specifications. The SSL V3 cipher specifications are used for the SSL V3 and TLS V1 protocols.

## **Related Topics**

**gsk\_attribute\_set\_buffer()**

**gsk\_environment\_open()**

**gsk\_secure\_socket\_open()**

## gsk\_attribute\_get\_cert\_info()

Returns certificate information following an SSL handshake.

### Format

```
#include <gskssl.h>

gsk_status gsk_attribute_get_cert_info (
    gsk_handle          soc_handle,
    GSK_CERT_ID        cert_id,
    gsk_cert_data_elem ** cert_data,
    int *              elem_count)
```

### Parameters

*soc\_handle*

Specifies the connection handle returned by the **gsk\_secure\_socket\_open()** routine.

*cert\_id*

Specifies the certificate identifier.

*cert\_data*

Returns the certificate data array. The **gsk\_free\_cert\_data()** routine should be called to release the array when the certificate information is no longer needed. A NULL address will be returned if no certificate information is available.

*elem\_count*

Returns the number of elements in the array of `gsk_cert_data_elem` structures.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ATTRIBUTE\_INVALID\_ID]

The certificate identifier is not valid.

#### [GSK\_ERR\_ASN]

Unable to decode certificate.

#### [GSK\_INSUFFICIENT\_STORAGE]

Insufficient storage is available.

#### [GSK\_INVALID\_HANDLE]

The connection handle is not valid.

#### [GSK\_INVALID\_STATE]

The connection is not initialized.

### Usage

The **gsk\_attribute\_get\_cert\_info()** routine returns information about certificates used in an SSL handshake. The connection must be in the initialized state. The certificate data address will be NULL if there is no certificate information available.

The following certificate identifiers are supported:

#### GSK\_LOCAL\_CERT\_INFO

Returns information about the local certificate.

#### GSK\_PARTNER\_CERT\_INFO

Returns information about the partner certificate.

## **gsk\_attribute\_get\_cert\_info()**

Each element of the certificate data array has an element identifier. The element identifiers used for a particular certificate will depend upon the contents of the certificate. The following element identifiers are currently provided:

### **CERT\_BODY\_BASE64**

Certificate body in Base64-encoded format

### **CERT\_BODY\_DER**

Certificate body in binary ASN.1 DER-encoded format

### **CERT\_COMMON\_NAME**

Subject common name (CN)

### **CERT\_COUNTRY**

Subject country (C)

### **CERT\_DN\_DER**

Subject distinguished name in binary ASN.1 DER-encoded format

### **CERT\_DN\_PRINTABLE**

Subject distinguished name as a printable character string

### **CERT\_DOMAIN\_COMPONENT**

Subject domain component (DC)

### **CERT\_EMAIL**

Subject e-mail address (EMAIL)

### **CERT\_ISSUER\_COMMON\_NAME**

Issuer common name (CN)

### **CERT\_ISSUER\_COUNTRY**

Issuer country (C)

### **CERT\_ISSUER\_DN\_DER**

Issuer distinguished name in binary ASN.1 DER-encoded format

### **CERT\_ISSUER\_DN\_PRINTABLE**

Issuer distinguished name as a printable character string

### **CERT\_ISSUER\_DOMAIN\_COMPONENT**

Issuer domain component (DC)

### **CERT\_ISSUER\_EMAIL**

Issuer e-mail address (EMAIL)

### **CERT\_ISSUER\_LOCALITY**

Issuer locality (L)

### **CERT\_ISSUER\_ORG**

Issuer organization (O)

### **CERT\_ISSUER\_ORG\_UNIT**

Issuer organizational unit (OU)

### **CERT\_ISSUER\_POSTAL\_CODE**

Issuer postal code (PC)

### **CERT\_ISSUER\_STATE\_OR\_PROVINCE**

Issuer state or province (SP)

### **CERT\_ISSUER\_STREET**

Issuer street (STREET)

### **CERT\_ISSUER\_SURNAME**

Issuer surname (SN)

**CERT\_ISSUER\_TITLE**

Issuer title (TITLE)

**CERT\_LOCALITY**

Subject locality (L)

**CERT\_ORG**

Subject organization (O)

**CERT\_ORG\_UNIT**

Subject organizational unit (OU)

**CERT\_POSTAL\_CODE**

Subject postal code (PC)

**CERT\_SERIAL\_NUMBER**

Certificate serial number

**CERT\_STATE\_OR\_PROVINCE**

Subject state or province (SP)

**CERT\_STREET**

Subject street (STREET)

**CERT\_SURNAME**

Subject surname (SN)

**CERT\_TITLE**

Subject title (TITLE)

The CERT\_BODY\_DER, CERT\_BODY\_BASE64, CERT\_DN\_DER, and CERT\_ISSUER\_DN\_DER elements are not null-terminated and the 'cert\_data\_' field must be used to get the element length. All of the other elements are null-terminated character strings and the 'cert\_data\_' field is the length of the string excluding the end-of-string delimiter.

**Related Topics****gsk\_secure\_socket\_init()****gsk\_free\_cert\_data()**

## **gsk\_attribute\_get\_data()**

---

### **gsk\_attribute\_get\_data()**

Returns information related to a certificate request.

#### **Format**

```
#include <gskssl.h>

gsk_status gsk_attribute_get_data (
    gsk_handle      soc_handle,
    GSK_DATA_ID    data_id,
    void **         data_ptr)
```

#### **Parameters**

*soc\_handle*

Specifies the connection handle returned by the **gsk\_secure\_socket\_open()** routine.

*data\_id*

Specifies the data identifier.

*data\_ptr*

Returns the address of the requested data. The address will be NULL if the requested data is not available.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

##### **[GSK\_ATTRIBUTE\_INVALID\_ID]**

The data identifier is not valid.

##### **[GSK\_ERR\_ASN]**

Unable to decode certification authority name.

##### **[GSK\_INSUFFICIENT\_STORAGE]**

Insufficient storage is available.

##### **[GSK\_INVALID\_HANDLE]**

The connection handle is not valid.

##### **[GSK\_INVALID\_STATE]**

The connection is not initialized.

#### **Usage**

The **gsk\_attribute\_get\_data()** routine returns information related to a certificate request. The server sends a certificate request to the client as part of the client authentication portion of the SSL handshake. The connection must be in the initialized state.

The following data identifiers are supported:

##### **GSK\_DATA\_ID\_SUPPORTED\_KEYS**

Returns a list of labels in the key database for certificates signed by a certification authority that is in the list provided by the server. A database entry will be included in the list only if it has both a certificate and a private key. The **gsk\_list\_free()** routine should be called to release the list when it is no longer needed.

##### **GSK\_DATA\_ID\_SERVER\_ISSUERS**

Returns a list of distinguished names of certification authorities provided by the server in the certificate request. The **gsk\_list\_free()** routine should be called to release the list when it is no longer needed.

| **Related Topics**

| `gsk_list_free()`

## **gsk\_attribute\_get\_enum()**

---

### **gsk\_attribute\_get\_enum()**

Gets an enumerated value.

#### **Format**

```
#include <gskssl.h>

gsk_status gsk_attribute_get_enum (
    gsk_handle          ssl_handle,
    GSK_ENUM_ID        enum_id,
    GSK_ENUM_VALUE *   enum_value)
```

#### **Parameters**

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*enum\_id*

Specifies the enumeration identifier.

*enum\_value*

Returns the enumeration value.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

##### **[GSK\_ATTRIBUTE\_INVALID\_ID]**

The enumeration identifier is not valid or cannot be used with the specified handle.

##### **[GSK\_INVALID\_HANDLE]**

The handle is not valid.

##### **[GSK\_INVALID\_STATE]**

The environment has been closed or the SSL connection has been established.

#### **Usage**

The **gsk\_attribute\_get\_enum()** routine will return an enumerated value for an SSL environment or an SSL connection.

The following enumeration identifiers are supported:

##### **GSK\_CLIENT\_AUTH\_TYPE**

Returns **GSK\_CLIENT\_AUTH\_FULL\_TYPE** if received certificates are validated by the System SSL runtime and **GSK\_CLIENT\_AUTH\_PASSTHRU\_TYPE** otherwise. **GSK\_CLIENT\_AUTH\_TYPE** can be specified only for an SSL environment.

##### **GSK\_PROTOCOL\_SSLV2**

Returns **GSK\_PROTOCOL\_SSLV2\_ON** if the SSL Version 2 protocol is enabled and **GSK\_PROTOCOL\_SSLV2\_OFF** if the SSL Version 2 protocol is not enabled. **GSK\_PROTOCOL\_SSLV2** can be specified for an SSL environment or an SSL connection.

##### **GSK\_PROTOCOL\_SSLV3**

Returns **GSK\_PROTOCOL\_SSLV3\_ON** if the SSL Version 3 protocol is enabled and **GSK\_PROTOCOL\_SSLV3\_OFF** if the SSL Version 3 protocol is not enabled. **GSK\_PROTOCOL\_SSLV3** can be specified for an SSL environment or an SSL connection.

##### **GSK\_PROTOCOL\_TLSV1**

Returns **GSK\_PROTOCOL\_TLSV1\_ON** if the TLS Version 1 protocol is enabled and

## **gsk\_attribute\_get\_enum()**

GSK\_PROTOCOL\_TLSV1\_OFF if the TLS Version 1 protocol is not enabled.  
GSK\_PROTOCOL\_TLSV1 can be specified for an SSL environment or an SSL connection.

### **GSK\_PROTOCOL\_USED**

Returns GSK\_PROTOCOL\_USED\_SSLV2 if the SSL Version 2 protocol was used to establish the connection, GSK\_PROTOCOL\_USED\_SSLV3 if the SSL Version 3 protocol was used to establish the connection, or GSK\_PROTOCOL\_USED\_TLSV1 if the TLS Version 1 protocol was used to establish the connection. GSK\_NULL will be returned if a connection has not been established. GSK\_PROTOCOL\_USED can be specified only for an SSL connection.

### **GSK\_SESSION\_TYPE**

Returns GSK\_CLIENT\_SESSION if the SSL handshake is to be performed as a client, GSK\_SERVER\_SESSION if the SSL handshake is to be performed as a server, or GSK\_SERVER\_SESSION\_WITH\_CL\_AUTH if the SSL handshake is to be performed as a server requiring client authentication. GSK\_SESSION\_TYPE can be specified for an SSL environment or an SSL connection.

### **GSK\_SID\_FIRST**

Returns GSK\_SID\_IS\_FIRST if a full SSL handshake was performed to establish the connection or GSK\_SID\_NOT\_FIRST if an existing session was used to establish the connection. GSK\_NULL will be returned if a connection has not been established. GSK\_SID\_FIRST can be specified only for an SSL connection.

### **GSK\_SYSPLEX\_SIDCACHE**

Returns GSK\_SYSPLEX\_SIDCACHE\_ON if sysplex session caching is enabled for this application or GSK\_SYSPLEX\_SIDCACHE\_OFF if sysplex session caching is not enabled. GSK\_SYSPLEX\_SIDCACHE can be specified only for an SSL environment.

## **Related Topics**

**gsk\_attribute\_set\_enum()**

**gsk\_environment\_open()**

**gsk\_secure\_socket\_open()**

## **gsk\_attribute\_get\_numeric\_value()**

---

### **gsk\_attribute\_get\_numeric\_value()**

Gets a numeric value.

#### **Format**

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

```
gsk_status gsk_attribute_get_numeric_value (
    gsk_handle    ssl_handle,
    GSK_NUM_ID   num_id,
    int *         num_value)
```

#### **Parameters**

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*num\_id*

Specifies the numeric identifier.

*num\_value*

Returns the numeric value.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

##### **[GSK\_ATTRIBUTE\_INVALID\_ID]**

The numeric identifier is not valid or cannot be used with the specified handle.

##### **[GSK\_INVALID\_HANDLE]**

The handle is not valid.

##### **[GSK\_INVALID\_STATE]**

The environment is closed.

#### **Usage**

The **gsk\_attribute\_get\_numeric\_value()** routine will return a numeric value for an SSL environment or an SSL connection.

The following numeric identifiers are supported:

##### **GSK\_CRL\_CACHE\_TIMEOUT**

Returns the CRL cache timeout. **GSK\_CRL\_CACHE\_TIMEOUT** can be specified only for an SSL environment.

##### **GSK\_FD**

Returns the socket descriptor used for network operations. **GSK\_FD** can be specified only for an SSL connection.

##### **GSK\_LDAP\_SERVER\_PORT**

Returns the LDAP server port. **GSK\_LDAP\_SERVER\_PORT** can be specified only for an SSL environment.

##### **GSK\_V2\_SESSION\_TIMEOUT**

Returns the SSL Version 2 session timeout. **GSK\_V2\_SESSION\_TIMEOUT** can be specified only for an SSL environment.

## **gsk\_attribute\_get\_numeric\_value()**

### **GSK\_V2\_SIDCACHE\_SIZE**

Returns the size of the SSL Version 2 session identifier cache. GSK\_V2\_SIDCACHE\_SIZE can be specified only for an SSL environment.

### **GSK\_V3\_SESSION\_TIMEOUT**

Returns the SSL Version 3 session timeout. GSK\_V3\_SESSION\_TIMEOUT can be specified only for an SSL environment.

### **GSK\_V3\_SIDCACHE\_SIZE**

Returns the size of the SSL Version 3 session identifier cache. GSK\_V3\_SIDCACHE\_SIZE can be specified only for an SSL environment.

## **Related Topics**

**gsk\_attribute\_set\_numeric\_value()**

**gsk\_environment\_open()**

**gsk\_secure\_socket\_open()**

## `gsk_attribute_set_buffer()`

---

### `gsk_attribute_set_buffer()`

Sets the value of an attribute buffer.

#### Format

```
#include <gskssl.h>

gsk_status gsk_attribute_set_buffer (
    gsk_handle      ssl_handle,
    GSK_BUF_ID     buffer_id,
    const char *    buffer_value,
    int             buffer_length)
```

#### Parameters

*ssl\_handle*

Specifies an SSL environment handle returned by `gsk_environment_open()` or an SSL connection handle returned by `gsk_secure_socket_open()`.

*buffer\_id*

Specifies the buffer identifier.

*buffer\_value*

Specifies the buffer value.

*buffer\_length*

Specifies the buffer length. Specify 0 for this parameter if the buffer value is a null-delimited character string.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

##### **[GSK\_ATTRIBUTE\_INVALID\_ID]**

The buffer identifier is not valid or cannot be used with the specified handle.

##### **[GSK\_ATTRIBUTE\_INVALID\_LENGTH]**

The buffer length is not valid.

##### **[GSK\_INSUFFICIENT\_STORAGE]**

Insufficient storage is available.

##### **[GSK\_INVALID\_HANDLE]**

The handle is not valid.

##### **[GSK\_INVALID\_STATE]**

The environment or connection is not in the open state.

#### Usage

The `gsk_attribute_set_buffer()` routine will set a buffer value in an SSL environment or an SSL connection. The environment or connection must be in the open state and not in the initialized state (that is, `gsk_environment_init()` or `gsk_secure_socket_init()` has not been called).

The following buffer identifiers are supported:

##### **GSK\_KEYRING\_FILE**

Specifies the name of the key database HFS file or the SAF key ring. A key database is used if a

## **gsk\_attribute\_set\_buffer()**

database password or stash file is defined using either an environment variable or the **gsk\_attribute\_set\_buffer()** routine. Otherwise a SAF key ring is used. GSK\_KEYRING\_FILE may be specified only for an SSL environment.

The SAF key ring name is specified as "userid/keyring". The current userid is used if the userid is omitted. The user must have READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by the user. The user must have UPDATE access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by another user. Note: Certificate private keys are not available when using a SAF key ring owned by another user.

### **GSK\_KEYRING\_LABEL**

Specifies the label of the key used to authenticate the application. The default key will be used if a key label is not specified. GSK\_KEYRING\_LABEL may be specified for an SSL environment or an SSL connection.

### **GSK\_KEYRING\_PW**

Specifies the password for the key database. GSK\_KEYRING\_PW may be specified only for an SSL environment.

### **GSK\_KEYRING\_STASH\_FILE**

Specifies the name of the key database password stash file. The stash file name always has an extension of ".sth" and the supplied name will be changed if it does not have the correct extension. The GSK\_KEYRING\_PW value will be used instead of the GSK\_KEYRING\_STASH value if it is also specified. GSK\_KEYRING\_STASH\_FILE may be specified only for an SSL environment.

### **GSK\_LDAP\_SERVER**

Specifies one or more blank-separated LDAP server host names. Each host name can contain an optional port number separated from the host name by a colon. GSK\_LDAP\_SERVER may be specified only for an SSL environment. The LDAP server is used to obtain CA certificates when validating a certificate and the local database does not contain the required certificate. The local database must contain the required certificates if no LDAP server is specified. Even when an LDAP server is used, root CA certificates must be found in the local database since the LDAP server is not a trusted data source. The LDAP server is also used to obtain certificate revocation lists.

### **GSK\_LDAP\_USER**

Specifies the distinguished name to use when connecting to the LDAP server. GSK\_LDAP\_USER may be specified only for an SSL environment.

### **GSK\_LDAP\_USER\_PW**

Specifies the password to use when connecting to the LDAP server. GSK\_LDAP\_USER\_PW may be specified only for an SSL environment.

### **GSK\_USER\_DATA**

Specifies the user data to be passed to SSL exit routines. The user data is copied to storage owned by the SSL runtime and the address of this storage is passed to the SSL exit routines. The application may alter this copy of the user data but may not free it. GSK\_USER\_DATA may be specified only for an SSL connection.

### **GSK\_V2\_CIPHER\_SPECS**

Specifies the SSL V2 cipher specifications as a string consisting of 1 or more 1-character values. GSK\_V2\_CIPHER\_SPECS may be specified for an SSL environment or an SSL connection. Refer to the description of **gsk\_environment\_open()** for a list of valid cipher specifications.

### **GSK\_V3\_CIPHER\_SPECS**

Specifies the SSL V3 cipher specifications as a string consisting of 1 or more 2-character values. GSK\_V3\_CIPHER\_SPECS may be specified for an SSL environment or an SSL connection. Refer to the description of **gsk\_environment\_open()** for a list of valid cipher specifications. The SSL V3 cipher specifications are used for the SSL V3 and TLS V1 protocols.

`gsk_attribute_set_buffer()`

## | **Related Topics**

| `gsk_attribute_get_buffer()`

| `gsk_environment_open()`

| `gsk_environment_init()`

| `gsk_secure_socket_open()`

| `gsk_secure_socket_init()`

## gsk\_attribute\_set\_callback()

Sets an SSL callback.

### Format

```
#include <gskssl.h>

gsk_status gsk_attribute_set_callback (
    gsk_handle          ssl_handle,
    GSK_CALLBACK_ID    callback_id,
    void *              callback)
```

### Parameters

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*callback\_id*

Specifies the callback identifier.

*callback*

Specifies the address of the callback parameter.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ATTRIBUTE\_INVALID\_ID]

The callback identifier is not valid or cannot be used with the specified handle.

#### [GSK\_ATTRIBUTE\_INVALID\_PARAMETER]

The attribute parameter value is not valid.

#### [GSK\_INVALID\_HANDLE]

The handle is not valid.

#### [GSK\_INVALID\_STATE]

The environment or connection is not in the open state.

### Usage

The **gsk\_attribute\_set\_callback()** routine establishes a callback to an application routine by the SSL runtime. A callback allows the application to replace the default routine used by the SSL runtime. The SSL environment or SSL connection must be in the open state and not in the initialized state (that is, **gsk\_environment\_init()** or **gsk\_secure\_socket\_init()** has not been called). The callback routine must use standard C linkage and not C++ linkage.

The following callback identifiers are supported:

#### GSK\_CLIENT\_CERT\_CALLBACK

Indicates the application is providing the routine to prompt a client user to select a certificate from a list during the client authentication process. The *callback* parameter is the address of this routine. The exit routine can obtain the user data address by calling the **gsk\_attribute\_get\_buffer()** routine. The **gsk\_attribute\_set\_buffer()** routine should be called to set the selected key label before returning from the callback routine. The function return value should be 0 if a key label has been set or **GSK\_ERR\_NO\_CERTIFICATE** if no client certificate is to be used. **GSK\_CLIENT\_CERT\_CALLBACK** can be specified only for an SSL environment.

## **gsk\_attribute\_set\_callback()**

Following is the prototype for the callback routine provided by the application. It shows the parameters passed to the application callback and the value returned by the callback.

```
int client_cert_callback (
    gsk_handle    soc_handle)
```

## **GSK\_IO\_CALLBACK**

Indicates the application is providing the routines to perform read, write, and control functions. The *callback* parameter is the address of a *gsk\_iocallback* structure. Each entry in the structure will override the corresponding SSL runtime routine. A NULL entry will cause the current callback routine to be used or the SSL runtime routine will be used if there is no callback routine.

GSK\_IO\_CALLBACK can be specified for an SSL environment or an SSL connection.

The routine specified by the *io\_read* entry is used to read data from the network. The *fd* parameter is the socket descriptor, the *buffer* parameter is the address of the data buffer, the *count* parameter is the buffer size, and the *user\_data* parameter is the user data address. The function return value should be 0 if the connection has been closed by the remote partner, -1 if an error is detected, or the number of bytes read from the network. The error code is returned in the *errno* runtime variable. The default routine uses the **recv()** library routine to read data from the network.

```
int io_read (
    int      fd,
    void *   buffer,
    int      count,
    char *   user_data)
```

The routine specified by the *io\_write* entry is used to write data to the network. The *fd* parameter is the socket descriptor, the *buffer* parameter is the address of the data buffer, the *count* parameter is the data length, and the *user\_data* parameter is the user data address. The function return value should be -1 if an error is detected or the number of bytes written to the network. The error code is returned in the *errno* runtime variable. The default routine uses the **send()** library routine to write data to the network.

```
int io_write (
    int      fd,
    void *   buffer,
    int      count,
    char *   user_data)
```

The routine specified by the *io\_getpeerid* entry is used to get the 32-bit network identifier for the remote partner. The *fd* parameter is the socket descriptor and the *user\_data* parameter is the user data address. However, the *io\_getpeerid* entry is deprecated and should not be used since it does not support IPv6 networks which use a 16-byte network identifier. Instead, the *io\_getpeername* entry should be used for both IPv4 and IPv6 networks. The *io\_getpeerid* entry will not be used if the *io\_getpeername* entry is not NULL.

```
unsigned long io_getpeerid (
    int      fd,
    char *   user_data)
```

The routine specified by the *io\_setsocketoptions* entry is used to set socket options. The *fd* parameter is the socket descriptor, the **cmd** parameter is the function to be performed, and the *user\_data* parameter is the user data address. The return value should be -1 if an error is detected and 0 otherwise. The error code is returned in the *errno* runtime variable. The **io\_setsocketoptions()** routine is called by the **gsk\_secure\_socket\_init()** routine before initiating the SSL handshake (GSK\_SET\_SOCKET\_STATE\_FOR\_HANDSHAKE) and again upon completion of the SSL handshake (GSK\_SET\_SOCKET\_STATE\_FOR\_READ\_WRITE). The default **io\_setsocketoptions()** routine puts the socket into blocking mode for GSK\_SET\_SOCKET\_STATE\_FOR\_HANDSHAKE and restores the original mode for GSK\_SET\_SOCKET\_STATE\_FOR\_READ\_WRITE.

```
int io_setsocketoptions (
    int      fd,
    int      cmd,
    char *   user_data)
```

The routine specified by the *io\_getpeername* entry is used to get the network identifier for the remote partner. The *fd* parameter is the socket descriptor, the *buffer* parameter is the address of the return buffer, the length parameter is the size of the return buffer, and the *user\_data* parameter is the user data address. Upon return, the *length* parameter should contain the actual length of the network identifier. The function return value should be -1 if an error is detected and 0 otherwise. The error code is returned in the *errno* runtime variable. The default routine uses the **getpeername()** library routine and returns the IP address of the remote partner (4 bytes for IPv4 and 16 bytes for IPv6) followed by the 2-byte port number.

```
int io_getpeername (
    int      fd,
    void *   buffer,
    int *    length,
    char *   user_data)
```

### GSK\_SID\_CACHE\_CALLBACK

Indicates the application is providing the routines to maintain the session identifier cache. The callback parameter is the address of a *gsk\_sidcache\_callback* structure. GSK\_SID\_CACHE\_CALLBACK can be specified only for an SSL environment and will be used only for SSL servers (the internal cache is always used for SSL clients).

The routine specified by the *Get* entry is called to retrieve an entry from the session identifier cache. The *session\_id* parameter is the session identifier, the *session\_id\_length* parameter is the length of the session identifier, and the *ssl\_version* parameter is the SSL protocol version number (GSK\_SSLVERSION\_V2 or GSK\_SSLVERSION\_V3). The function return value is the address of the session data buffer or NULL if an error is detected. The *FreeDataBuffer* routine will be called to release the session data buffer when it is no longer needed by the SSL runtime.

```
gsk_data_buffer * Get (
    const unsigned char * session_id,
    unsigned int         session_id_length,
    gsk_sslversion       ssl_version)
```

The routine specified by the *Put* entry is called to store an entry in the session identifier cache. The *ssl\_session\_data* parameter is the session data, the *session\_id* parameter is the session identifier, the *session\_id\_length* parameter is the length of the session identifier, and the *ssl\_version* parameter is the SSL protocol version number (GSK\_SSLVERSION\_V2 or GSK\_SSLVERSION\_V3). The function return value is ignored and can be a NULL address. The callback routine must make its own copy of the session data since the SSL structure will be released when the connection is closed.

```
gsk_data_buffer * Put (
    gsk_data_buffer * ssl_session_data,
    const unsigned char * session_id,
    unsigned int         session_id_length,
    gsk_sslversion       ssl_version)
```

The routine specified by the *Delete* entry is called to remove an entry from the session identifier cache. The *session\_id* parameter is the session identifier, the *session\_id\_length* parameter is the length of the session identifier, and the *ssl\_version* parameter is the SSL protocol version number (GSK\_SSLVERSION\_V2 or GSK\_SSLVERSION\_V3).

```
void Delete (
    const unsigned char * session_id,
    unsigned int         session_id_length,
    gsk_sslversion       ssl_version)
```

## **gsk\_attribute\_set\_callback()**

The routine specified by the *FreeDataBuffer* entry is called to release the data buffer returned by the *Get* routine.

```
void FreeDataBuffer (
    gsk_data_buffer *    ssl_session_data)
```

## gsk\_attribute\_set\_enum()

Sets an enumerated value.

### Format

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

```
gsk_status gsk_attribute_set_enum (
    gsk_handle          ssl_handle,
    GSK_ENUM_ID        enum_id,
    GSK_ENUM_VALUE     enum_value)
```

### Parameters

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*enum\_id*

Specifies the enumeration identifier.

*enum\_value*

Specifies the enumeration value.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ATTRIBUTE\_INVALID\_ID]

The enumeration identifier is not valid or cannot be used with the specified handle.

#### [GSK\_INVALID\_HANDLE]

The handle is not valid.

#### [GSK\_INVALID\_STATE]

The environment or connection is not in the open state.

### Usage

The **gsk\_attribute\_set\_enum()** routine will set an enumerated value for an SSL environment or an SSL connection. The environment or connection must be in the open state and not in the initialized state (that is, **gsk\_environment\_init()** or **gsk\_secure\_socket\_init()** has not been called).

The following enumeration identifiers are supported:

#### GSK\_CLIENT\_AUTH\_TYPE

Specifies **GSK\_CLIENT\_AUTH\_FULL\_TYPE** to validate client certificates. If a certificate is not valid, the connection is not started and an error code is returned by the **gsk\_secure\_socket\_init()** routine. If an LDAP server is specified, the LDAP server is queried for CA certificates and certificate revocation lists. If the LDAP server is not available, only local validation will be performed. If no client certificate is received, the connection is successful. The application can check for this case by calling the **gsk\_attribute\_get\_cert\_info()** routine and checking for a NULL return address.

Specify **GSK\_CLIENT\_AUTH\_PASSTHRU\_TYPE** to bypass client certificate validation. The application can retrieve the certificate by calling the **gsk\_attribute\_get\_cert\_info()** routine.

**GSK\_CLIENT\_AUTH\_TYPE** can be specified only for an SSL environment.

#### GSK\_PROTOCOL\_SSLV2

Specifies **GSK\_PROTOCOL\_SSLV2\_ON** to enable the SSL Version 2 protocol or

## **gsk\_attribute\_set\_enum()**

GSK\_PROTOCOL\_SSLV2\_OFF to disable the SSL Version 2 protocol. The SSL V2 protocol should be disabled whenever possible since the SSL V3 protocol provides significant security enhancements.

GSK\_PROTOCOL\_SSLV2 can be specified for an SSL environment or an SSL connection.

## **GSK\_PROTOCOL\_SSLV3**

Specifies GSK\_PROTOCOL\_SSLV3\_ON to enable the SSL Version 3 protocol or GSK\_PROTOCOL\_SSLV3\_OFF to disable the SSL Version 3 protocol.

GSK\_PROTOCOL\_SSLV3 can be specified for an SSL environment or an SSL connection.

## **GSK\_PROTOCOL\_TLSV1**

Specifies GSK\_PROTOCOL\_TLSV1\_ON to enable the TLS Version 1 protocol or GSK\_PROTOCOL\_TLSV1\_OFF to disable the SSL Version 3 protocol.

GSK\_PROTOCOL\_TLSV1 can be specified for an SSL environment or an SSL connection.

## **GSK\_SESSION\_TYPE**

Specifies GSK\_CLIENT\_SESSION to perform the SSL handshake as a client, GSK\_SERVER\_SESSION to perform the SSL handshake as a server, or GSK\_SERVER\_SESSION\_WITH\_CL\_AUTH to perform the SSL handshake as a server requiring client authentication.

GSK\_SESSION\_TYPE can be specified for an SSL environment or an SSL connection.

## **GSK\_SYSPLEX\_SIDCACHE**

Returns GSK\_SYSPLEX\_SIDCACHE\_ON if sysplex session caching is enabled for this application or GSK\_SYSPLEX\_SIDCACHE\_OFF if sysplex session caching is not enabled.

GSK\_SYSPLEX\_SIDCACHE can be specified only for an SSL environment.

## **Related Topics**

**gsk\_attribute\_get\_enum()**

**gsk\_environment\_open()**

**gsk\_environment\_init()**

**gsk\_secure\_socket\_open()**

**gsk\_secure\_socket\_init()**

## gsk\_attribute\_set\_numeric\_value()

Sets a numeric value.

### Format

```
#include <gskssl.h>

gsk_status gsk_attribute_set_numeric_value (
    gsk_handle    ssl_handle,
    GSK_NUM_ID    num_id,
    int           num_value)
```

### Parameters

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*num\_id*

Specifies the numeric identifier.

*num\_value*

Specifies the numeric value.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ATTRIBUTE\_INVALID\_ID]

The numeric identifier is not valid or cannot be used with the specified handle.

#### [GSK\_ATTRIBUTE\_INVALID\_NUMERIC\_VALUE]

The numeric value is not within the valid range.

#### [GSK\_INVALID\_HANDLE]

The handle is not valid.

#### [GSK\_INVALID\_STATE]

The environment or connection is not in the open state.

### Usage

The **gsk\_attribute\_set\_numeric\_value()** routine will set a numeric value for an SSL environment or an SSL connection. The environment or connection must be in the open state and not in the initialized state (that is, **gsk\_environment\_init()** or **gsk\_secure\_socket\_init()** has not been called).

The following numeric identifiers are supported:

#### **GSK\_CRL\_CACHE\_TIMEOUT**

Sets the CRL cache timeout. This is the number of hours that a cached CRL will remain valid. The range is 0-720 and defaults to 24. A value of 0 will disable CRL caching.

**GSK\_CRL\_CACHE\_TIMEOUT** can be specified only for an SSL environment.

#### **GSK\_FD**

Sets the socket descriptor for network operations. **GSK\_FD** can be specified only for an SSL connection. The socket must not be closed until the **gsk\_secure\_socket\_close()** routine has been called to terminate the secure connection.

#### **GSK\_LDAP\_SERVER\_PORT**

Sets the LDAP server port. The port must be between 1 and 65535. Port 389 will be used if no

### **gsk\_attribute\_set\_numeric\_value()**

LDAP server port is set. GSK\_LDAP\_SERVER\_PORT can be specified only for an SSL environment. GSK\_LDAP\_SERVER\_PORT can be specified only for an SSL environment.

### **GSK\_V2\_SESSION\_TIMEOUT**

Sets the SSL Version 2 session timeout. This is the number of seconds until an SSL V2 session identifier expires. The range is 0-100 and defaults to 100. System SSL will remember SSL V2 session identifiers for this amount of time. This reduces the amount of data exchanged during the SSL handshake when a complete initial handshake has already been performed. Session identifiers will not be remembered if a value of 0 is specified. GSK\_V2\_SESSION\_TIMEOUT can be specified only for an SSL environment.

### **GSK\_V2\_SIDCACHE\_SIZE**

Sets the size of the SSL Version 2 session identifier cache. The oldest entry will be removed when the cache is full in order to add a new entry. The range is 0-32000 and defaults to 256. Session identifiers will not be remembered if a value of 0 is specified. GSK\_V2\_SIDCACHE\_SIZE can be specified only for an SSL environment.

### **GSK\_V3\_SESSION\_TIMEOUT**

Sets the session timeout for the SSL V3 and TLS V1 protocols. This is the number of seconds until an SSL V3 session identifier expires. The range is 0-86400 and defaults to 86400. System SSL will remember session identifiers for this amount of time. This reduces the amount of data exchanged during the SSL handshake when a complete initial handshake has already been performed. Session identifiers will not be remembered if a value of 0 is specified. GSK\_V3\_SESSION\_TIMEOUT can be specified only for an SSL environment.

### **GSK\_V3\_SIDCACHE\_SIZE**

Sets the size of the SSL Version 3 session identifier cache. The oldest entry will be removed when the cache is full in order to add a new entry. The range is 0-64000 and defaults to 512. Session identifiers will not be remembered if a value of 0 is specified. The SSL V3 session cache is used for the SSL V3 and TLS V1 protocols. GSK\_V3\_SIDCACHE\_SIZE can be specified only for an SSL environment.

## **Related Topics**

**gsk\_attribute\_get\_numeric\_value()**

**gsk\_environment\_open()**

**gsk\_environment\_init()**

**gsk\_secure\_socket\_init()**

**gsk\_secure\_socket\_open()**

---

## gsk\_environment\_close()

Closes a SSL environment.

### Format

```
#include <gskssl.h>

gsk_status gsk_environment_close (
    gsk_handle *    env_handle)
```

### Parameters

*env\_handle*

Specifies the SSL environment handle returned by the **gsk\_environment\_open()** routine. The environment handle will be set to NULL upon completion.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

**[GSK\_INVALID\_HANDLE]**

The environment handle is not valid.

**[GSK\_INVALID\_STATE]**

The environment is already closed.

### Usage

The **gsk\_environment\_close()** routine will close an environment created by the **gsk\_environment\_open()** routine. The storage allocated for the environment will not be released until all connections created using the environment have been closed. The SSL environment cannot be used to create new connections upon completion of the close.

### Related Topics

**gsk\_environment\_open()**

**gsk\_environment\_init()**

**gsk\_secure\_socket\_init()**

**gsk\_secure\_socket\_close()**

## `gsk_environment_init()`

---

### `gsk_environment_init()`

Initializes a SSL environment.

#### Format

```
#include <gskssl.h>

gsk_status gsk_environment_init (
    gsk_handle    env_handle)
```

#### Parameters

*env\_handle*

Specifies the SSL environment handle returned by the `gsk_environment_open()` routine.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

##### **[GSK\_CERTIFICATE\_NOT\_AVAILABLE]**

The key database or key ring does not contain any certificates.

##### **[GSK\_ERR\_BAD\_KEYFILE\_PASSWORD]**

The key database password is not correct.

##### **[GSK\_ERR\_LDAP]**

Unable to initialize the LDAP client.

##### **[GSK\_ERR\_LDAP\_NOT\_AVAILABLE]**

The LDAP server is not available.

##### **[GSK\_ERR\_PERMISSION\_DENIED]**

Not authorized to access key database or key ring.

##### **[GSK\_INSUFFICIENT\_STORAGE]**

Insufficient storage is available.

##### **[GSK\_INVALID\_HANDLE]**

The environment handle is not valid.

##### **[GSK\_INVALID\_STATE]**

The environment is not in the open state.

##### **[GSK\_KEYFILE\_INVALID\_FORMAT]**

The database is not a key database.

##### **[GSK\_KEYFILE\_IO\_ERR]**

An input/output error occurred while reading the key database or key ring.

##### **[GSK\_KEYFILE\_PASSWORD\_EXPIRED]**

The key database password is expired.

##### **[GSK\_KEYRING\_OPEN\_ERROR]**

Unable to open the key database or key ring.

##### **[GSK\_NO\_KEYFILE\_PASSWORD]**

The key database password is not available.

##### **[GSK\_RSA\_TEMP\_KEY\_PAIR]**

Unable to generate temporary RSA public/private key pair.

**Usage**

The **gsk\_environment\_init()** routine initializes an SSL environment created by the **gsk\_environment\_open()** routine. After the SSL environment has been initialized, it can be used to create one or more SSL connections by calling the **gsk\_secure\_socket\_open()** routine. The **gsk\_environment\_close()** routine should be called to close the environment when it is no longer needed. The **gsk\_environment\_close()** routine should also be called if an error is returned by the **gsk\_environment\_init()** routine.

**Related Topics**

- | **gsk\_environment\_open()**
- | **gsk\_environment\_close()**
- | **gsk\_secure\_socket\_open()**

## `gsk_environment_open()`

---

### `gsk_environment_open()`

Creates a SSL environment.

#### Format

```
#include <gskssl.h>

gsk_status gsk_environment_open (
    gsk_handle *    env_handle)
```

#### Parameters

*env\_handle*

Returns the handle for the environment. The application should call the **`gsk_environment_close()`** routine to release the environment when it is no longer needed.

#### Results

The function return value will be 0 (**`GSK_OK`**) if no error is detected. Otherwise, it will be one of the return codes listed in the **`gskssl.h`** include file. The following are some common errors:

##### **`[GSK_ATTRIBUTE_INVALID_ENUMERATION]`**

The value of an environment variable is not valid.

##### **`[GSK_ATTRIBUTE_INVALID_LENGTH]`**

The length of an environment variable value is not valid.

##### **`[GSK_ATTRIBUTE_INVALID_NUMERIC_VALUE]`**

The value of an environment variable is not valid.

##### **`[GSK_INSUFFICIENT_STORAGE]`**

Insufficient storage is available.

#### Usage

The **`gsk_environment_open()`** routine creates an SSL environment. The environment will be initialized with default values and then any SSL environment variables will be processed. These values can be changed by the application using the appropriate **`gsk_attribute_set_*`** routines. The **`gsk_environment_init()`** routine should then be called to initialize the SSL environment. This environment can then be used to establish one or more SSL connections.

The following default values are set:

- SSL V2, SSL V3, and TLS V1 are enabled
- The connection type is set to CLIENT
- The SSL V2 connection timeout is set to 100 seconds
- The SSL V3 connection timeout is set to 86400 seconds
- The SSL V2 cache size is set to 256
- The SSL V3 cache size is set to 512
- The sysplex session cache is disabled.
- The SSL V2 cipher specification is set to "713642" if domestic encryption is enabled and "642" otherwise
- The SSL V3 cipher specification is set to "0504352F0A090306020100" if domestic encryption is enabled and "090306020100" otherwise
- The default key will be used
- No LDAP server will be used

- The default callback routines will be used

The following SSL V2 cipher specifications are supported:

- "1" = 128-bit RC4 encryption with MD5 message authentication (128-bit secret key)
- "2" = 128-bit RC4 export encryption with MD5 message authentication (40-bit secret key)
- "3" = 128-bit RC2 encryption with MD5 message authentication (128-bit secret key)
- "4" = 128-bit RC2 export encryption with MD5 message authentication (40-bit secret key)
- "6" = 56-bit DES encryption with MD5 message authentication (56-bit secret key)
- "7" = 168-bit Triple DES encryption with MD5 message authentication (168-bit secret key)

The following SSL V3 cipher specifications are supported:

- "00" = No encryption or message authentication
- "01" = No encryption with MD5 message authentication
- "02" = No encryption with SHA-1 message authentication
- "03" = 40-bit RC4 encryption with MD5 message authentication and RSA key exchange
- "04" = 128-bit RC4 encryption with MD5 message authentication and RSA key exchange
- "05" = 128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange
- "06" = 40-bit RC2 encryption with MD5 message authentication and RSA key exchange
- "09" = 56-bit DES encryption with SHA-1 message authentication and RSA key exchange
- "0A" = 168-bit Triple DES encryption with SHA-1 message authentication and RSA key exchange
- "2F" = 128-bit AES encryption with SHA-1 message authentication and RSA key exchange
- "35" = 256-bit AES encryption with SHA-1 message authentication and RSA key exchange

The following environment variables are processed:

#### **GSK\_KEY\_LABEL**

Specifies the label of the key used to authenticate the application. The default key will be used if a key label is not specified.

#### **GSK\_KEYRING\_FILE**

Specifies the name of the key database HFS file or the SAF key ring. A key database is used if the GSK\_KEYRING\_PW or GSK\_KEYRING\_STASH environment variable is also specified. Otherwise a SAF key ring is used.

The SAF key ring name is specified as "userid/keyring". The current userid is used if the userid is omitted. The user must have READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by the user. The user must have UPDATE access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by another user. Note: Certificate private keys are not available when using a SAF key ring owned by another user.

#### **GSK\_KEYRING\_PW**

Specifies the password for the key database.

#### **GSK\_KEYRING\_STASH**

Specifies the name of the key database password stash file. The stash file name always has an extension of ".sth" and the supplied name will be changed if it does not have the correct extension. The GSK\_KEYRING\_PW environment variable will be used instead of the GSK\_KEYRING\_STASH environment variable if it is also specified.

#### **GSK\_LDAP\_SERVER**

Specifies one or more blank-separated LDAP server host names. Each host name can contain an optional port number separated from the host name by a colon. The LDAP server is used to obtain CA certificates when validating a certificate and the local database does not contain the required certificate. The local database must contain the required certificates if no LDAP server is specified.

## **gsk\_environment\_open()**

Even when an LDAP server is used, root CA certificates must be found in the local database since the LDAP server is not a trusted data source. The LDAP server is also used to obtain certificate revocation lists.

### **[GSK\_LDAP\_PASSWORD]**

Specifies the password to use when connecting to the LDAP server.

### **[GSK\_LDAP\_PORT]**

Specifies the LDAP server port. Port 389 will be used if no LDAP server port is specified.

### **GSK\_LDAP\_USER**

Specifies the distinguished name to use when connecting to the LDAP server.

### **GSK\_PROTOCOL\_SSLV2**

Specifies whether the SSL V2 protocol is supported. A value of "0", "OFF" or "DISABLED" will disable the SSL V2 protocol while a value of "1", "ON" or "ENABLED" will enable the SSL V2 protocol. The SSL V2 protocol should be disabled whenever possible since the SSL V3 protocol provides significant security enhancements.

### **GSK\_PROTOCOL\_SSLV3**

Specifies whether the SSL V3 protocol is supported. A value of "0", "OFF" or "DISABLED" will disable the SSL V3 protocol while a value of "1", "ON" or "ENABLED" will enable the SSL V3 protocol.

### **GSK\_PROTOCOL\_TLSV1**

Specifies whether the TLS V1 protocol is supported. A value of "0", "OFF" or "DISABLED" will disable the TLS V1 protocol while a value of "1", "ON" or "ENABLED" will enable the TLS V1 protocol. The TLS V1 protocol uses the same session cache and cipher specifications as the SSL V3 protocol.

### **GSK\_SYSPLEX\_SIDCACHE**

Specifies whether sysplex session caching is supported for this application. A value of "0", "OFF" or "DISABLED" will disable sysplex session caching while a value of "1", "ON" or "ENABLED" will enable sysplex session caching.

### **GSK\_V2\_CIPHER\_SPECS**

Specifies the SSL V2 cipher specifications in order of preference as a null-terminated string consisting of 1 or more 1-character values. Valid cipher specifications that are not supported due to the installed cryptographic level will be skipped when the connection is initialized.

### **GSK\_V2\_SESSION\_TIMEOUT**

Specifies the session timeout value in seconds for the SSL V2 protocol. The valid timeout values are 0 through 100 and defaults to 100.

### **GSK\_V2\_SIDCACHE\_SIZE**

Specifies the number of session identifiers that can be contained in the SSL V2 cache. The valid cache sizes are 0 through 32000 and defaults to 256. The SSL V2 cache will be disabled if 0 is specified.

### **GSK\_V3\_CIPHER\_SPECS**

Specifies the SSL V3 cipher specifications in order of preference as a null-terminated string consisting of 1 or more 2-character values. The SSL V3 cipher specifications are used for both the SSL V3 and the TLS V1 protocols. Valid cipher specifications that are not supported due to the installed cryptographic level will be skipped when the connection is initialized.

### **GSK\_V3\_SESSION\_TIMEOUT**

Specifies the session timeout value in seconds for the SSL V3 and TLS V1 protocols. The valid timeout values are 0 through 86400 and defaults to 86400.

### **GSK\_V3\_SIDCACHE\_SIZE**

Specifies the number of session identifiers that can be contained in the SSL V3 cache. The valid

## **gsk\_environment\_open()**

cache sizes are 0 through 64000 and defaults to 512. The SSL V3 cache will be disabled if 0 is specified. The SSL V3 cache is used for the SSL V3 and TLS V1 protocols.

### **Related Topics**

**gsk\_environment\_init()**

**gsk\_environment\_close()**

**gsk\_free\_cert\_data()**

---

## **gsk\_free\_cert\_data()**

Releases the storage allocated for a certificate data array.

### **Format**

```
#include <gskssl.h>

void gsk_free_cert_data (
    gsk_cert_data_elem *   cert_data,
    int                    elem_count)
```

### **Parameters**

*cert\_data*

Specifies the certificate data array to be released.

*elem\_count*

Specifies the number of elements in the certificate data array.

### **Usage**

The **gsk\_free\_cert\_data()** routine will release the storage allocated for an array of certificate data elements.

### **Related Topics**

**gsk\_attribute\_get\_cert\_info()**

**gsk\_get\_cert\_by\_label()**

## gsk\_get\_cert\_by\_label()

Gets certificate information for a record label.

### Format

```
#include <gskssl.h>

gsk_status gsk_get_cert_by_label (
    gsk_handle          ssl_handle,
    const char *       record_label,
    gsk_cert_data_elem ** cert_data,
    int *              elem_count)
```

### Parameters

*ssl\_handle*

Specifies an SSL environment handle returned by **gsk\_environment\_open()** or an SSL connection handle returned by **gsk\_secure\_socket\_open()**.

*record\_label*

Specifies the record label for the certificate.

*cert\_data*

Returns the certificate data array. The **gsk\_free\_cert\_data()** routine should be called to release the array when the certificate information is no longer needed.

*elem\_count*

Returns the number of elements in the array of `gsk_cert_data_elem` structures.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ERR\_ASN]

Unable to decode certificate.

#### [GSK\_INSUFFICIENT\_STORAGE]

Insufficient storage is available.

#### [GSK\_INVALID\_HANDLE]

The handle is not valid.

#### [GSK\_KEY\_LABEL\_NOT\_FOUND]

The key record is not found.

### Usage

The **gsk\_get\_cert\_by\_label()** routine returns certificate information for a record label. The supplied handle can be for an SSL environment or an SSL connection.

Each element of the certificate data array has an element identifier. The element identifiers used for a particular certificate will depend upon the contents of the certificate. The following element identifiers are currently provided:

#### CERT\_BODY\_BASE64

Certificate body in Base64-encoded format

#### CERT\_BODY\_DER

Certificate body in binary ASN.1 DER-encoded format

## **gsk\_get\_cert\_by\_label()**

### **CERT\_COMMON\_NAME**

Subject common name (CN)

### **CERT\_COUNTRY**

Subject country (C)

### **CERT\_DN\_DER**

Subject distinguished name in binary ASN.1 DER-encoded format

### **CERT\_DN\_PRINTABLE**

Subject distinguished name as a printable character string

### **CERT\_DOMAIN\_COMPONENT**

Subject domain component (DC)

### **CERT\_EMAIL**

Subject e-mail address (EMAIL)

### **CERT\_ISSUER\_COMMON\_NAME**

Issuer common name (CN)

### **CERT\_ISSUER\_COUNTRY**

Issuer country (C)

### **CERT\_ISSUER\_DN\_DER**

Issuer distinguished name in binary ASN.1 DER-encoded format

### **CERT\_ISSUER\_DN\_PRINTABLE**

Issuer distinguished name as a printable character string

### **CERT\_ISSUER\_DOMAIN\_COMPONENT**

Issuer domain component (DC)

### **CERT\_ISSUER\_EMAIL**

Issuer e-mail address (EMAIL)

### **CERT\_ISSUER\_LOCALITY**

Issuer locality (L)

### **CERT\_ISSUER\_ORG**

Issuer organization (O)

### **CERT\_ISSUER\_ORG\_UNIT**

Issuer organizational unit (OU)

### **CERT\_ISSUER\_POSTAL\_CODE**

Issuer postal code (PC)

### **CERT\_ISSUER\_STATE\_OR\_PROVINCE**

Issuer state or province (SP)

### **CERT\_ISSUER\_STREET**

Issuer street (STREET)

### **CERT\_ISSUER\_SURNAME**

Issuer surname (SN)

### **CERT\_ISSUER\_TITLE**

Issuer title (TITLE)

### **CERT\_LOCALITY**

Subject locality (L)

### **CERT\_ORG**

Subject organization (O)

| **CERT\_ORG\_UNIT**

| Subject organizational unit (OU)

| **CERT\_POSTAL\_CODE**

| Subject postal code (PC)

| **CERT\_SERIAL\_NUMBER**

| Certificate serial number

| **CERT\_STATE\_OR\_PROVINCE**

| Subject state or province (SP)

| **CERT\_STREET**

| Subject street (STREET)

| **CERT\_SURNAME**

| Subject surname (SN)

| **CERT\_TITLE**

| Subject title (TITLE)

| The CERT\_BODY\_DER, CERT\_BODY\_BASE64, CERT\_DN\_DER, and CERT\_ISSUER\_DN\_DER elements are not null-terminated and the 'cert\_data\_' field must be used to get the element length. All of the other elements are null-terminated character strings and the 'cert\_data\_' field is the length of the string excluding the string delimiter.

| **Related Topics**

| **gsk\_environment\_init()**

| **gsk\_secure\_socket\_init()**

**gsk\_get\_cipher\_suites()**

---

## **gsk\_get\_cipher\_suites()**

Returns the available SSL cipher suites.

### **Format**

```
#include <gskssl.h>

void gsk_get_cipher_suites (
    gsk_cipher_suites *    cipher_suites)
```

### **Parameters**

*cipher\_suites*

Returns the runtime version, release, security level, and cipher suites.

### **Usage**

The **gsk\_get\_cipher\_suites()** routine returns the System SSL runtime version, release, security level, and available cipher suites. The current System SSL runtime is Version 3 Release 14. The cipher suites are static null-terminated character strings which must not be modified or freed by the application.

## gsk\_get\_update()

Checks for a key database or key ring update.

### Format

```
#include <gskssl.h>

gsk_status gsk_get_update (
    gsk_handle     env_handle,
    long *         update_flag)
```

### Parameters

*env\_handle*

Specifies the SSL environment handle returned by the **gsk\_environment\_open()** routine.

*update\_flag*

Returns 1 if the key database or key ring has been updated or 0 if it has not been updated.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_INVALID\_HANDLE]

The environment handle is not valid.

#### [GSK\_INVALID\_STATE]

The environment is not in the initialized state.

#### [GSK\_KEYRING\_OPEN\_ERROR]

The key ring cannot be accessed.

### Usage

The **gsk\_get\_update()** routine tests if the key database or key ring associated with the SSL environment has been updated since the last time that **gsk\_get\_update()** was called or since the environment was initialized if **gsk\_get\_update()** has not been called yet. If an update has occurred, the application can close the current environment and then create a new environment to pick up the updates.

### Related Topics

**gsk\_environment\_open()**

## **gsk\_list\_free()**

---

### **gsk\_list\_free()**

Releases storage allocated for a list.

### **Format**

```
#include <gskssl.h>

void gsk_list_free (
    gsk_list *    list)
```

### **Parameters**

*list* Specifies the list to be released.

### **Usage**

The **gsk\_list\_free()** routine releases storage allocated for a list. This includes the `gsk_list` structure itself and all `gsk_list` structures anchored by the structure passed on the function call.

### **Related Topics**

**gsk\_attribute\_get\_data()**

## gsk\_secure\_socket\_close()

Closes a secure socket connection.

### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_close (
    gsk_handle *    soc_handle)
```

### Parameters

*soc\_handle*

Specifies the connection handle returned by the **gsk\_secure\_socket\_open()** routine. The connection handle will be set to NULL upon completion.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_CONNECTION\_ACTIVE]

The connection has an active read or write request.

#### [GSK\_INVALID\_HANDLE]

The connection handle is not valid.

#### [GSK\_WOULD\_BLOCK\_WRITE]

An attempt to write pending data failed with EWOULDBLOCK.

### Usage

The **gsk\_secure\_socket\_close()** routine will close a secure socket connection created by the **gsk\_secure\_socket\_open()** routine. The socket itself is not closed (the application is responsible for closing the socket). The connection can no longer be used for secure communications after calling the **gsk\_secure\_socket\_close()** routine.

The **gsk\_secure\_socket\_close()** routine can return GSK\_WOULD\_BLOCK\_WRITE if the socket is in non-blocking mode and there is pending write data. The connection is not closed in this case and the application should call **gsk\_secure\_socket\_close()** again when the socket is ready to accept a write request.

### Related Topics

**gsk\_secure\_socket\_open()**

**gsk\_secure\_socket\_init()**

**gsk\_secure\_socket\_read()**

**gsk\_secure\_socket\_write()**

## `gsk_secure_socket_init()`

---

### `gsk_secure_socket_init()`

Initializes a secure socket connection.

### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_init(
    gsk_handle    soc_handle)
```

### Parameters

*soc\_handle*

Specifies the connection handle returned by the `gsk_secure_socket_open()` routine.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

#### **[GSK\_ERR\_BAD\_CERT]**

Certificate is not valid.

#### **[GSK\_ERR\_BAD\_DATE]**

Certificate is not valid yet or is expired.

#### **[GSK\_ERR\_BAD\_KEYFILE\_LABEL]**

The specified key is not found in the key database or the key is not trusted.

#### **[GSK\_ERR\_BAD\_MAC]**

Message verification failed.

#### **[GSK\_ERR\_BAD\_MESSAGE]**

Incorrectly-formatted message received from peer application.

#### **[GSK\_ERR\_BAD\_PEER]**

Peer application has violated the SSL protocol.

#### **[GSK\_ERR\_CERT\_VALIDATION]**

Certificate validation error.

#### **[GSK\_ERR\_CERTIFICATE\_REVOKED]**

Peer certificate is revoked.

#### **[GSK\_ERR\_CRYPT0]**

Cryptographic error detected.

#### **[GSK\_ERR\_INCOMPATIBLE\_KEY]**

Certificate key is not compatible with cipher suite.

#### **[GSK\_ERR\_IO]**

I/O error communicating with peer application.

#### **[GSK\_ERR\_LDAP]**

An LDAP error is detected.

#### **[GSK\_ERR\_LDAP\_NOT\_AVAILABLE]**

The LDAP server is not available.

#### **[GSK\_ERR\_NO\_CIPHERS]**

No cipher specifications.

**[GSK\_ERR\_SELF\_SIGNED]**

A self-signed certificate cannot be validated.

**[GSK\_ERR\_SOCKET\_CLOSED]**

Socket connection closed by peer application.

**[GSK\_ERR\_UNKNOWN\_CA]**

A certification authority certificate is missing.

**[GSK\_ERR\_UNSUPPORTED\_CERTIFICATE\_TYPE]**

The certificate type is not supported by System SSL.

**[GSK\_INSUFFICIENT\_STORAGE]**

Insufficient storage is available.

**[GSK\_INVALID\_HANDLE]**

The connection handle is not valid.

**[GSK\_INVALID\_STATE]**

The connection is not in the open state or a previous initialization request has failed.

**[GSK\_WOULD\_BLOCK\_READ]**

An attempt to read a handshake message failed with EWOULDBLOCK.

**[GSK\_WOULD\_BLOCK\_WRITE]**

An attempt to write a handshake message failed with EWOULDBLOCK.

## Usage

The **gsk\_secure\_socket\_init()** routine initializes a secure socket connection created by the **gsk\_secure\_socket\_open()** routine. After the connection has been initialized, it can be used for secure data transmission using the **gsk\_secure\_socket\_read()** and **gsk\_secure\_socket\_write()** routines. The **gsk\_secure\_socket\_close()** routine should be called to close the connection when it is no longer needed. The **gsk\_secure\_socket\_close()** routine should also be called if an error is returned by the **gsk\_secure\_socket\_init()** routine.

Before calling the **gsk\_secure\_socket\_init()** routine, the application must create a connected socket and store the socket descriptor in the SSL connection by calling the **gsk\_attribute\_set\_numeric\_value()** routine. For a client, this means calling the **socket()** and **connect()** routines. For a server, this means calling the **socket()**, **listen()**, and **accept()** routines. However, SSL does not require the use of TCP/IP for the communications layer. The socket descriptor can be any integer value which is meaningful to the application. The application must provide its own socket routines if it is not using TCP/IP by calling the **gsk\_attribute\_set\_callback()** routine.

An SSL handshake is performed as part of the processing of the **gsk\_secure\_socket\_init()** routine. This establishes the server identity and optionally the client identity. It also negotiates the cryptographic parameters to be used for the connection. The client and server will attempt to use the highest available protocol version as determined by the intersection of the enabled protocol versions for the client and the server. Thus, TLS V1 will be used if it is enabled on both the client and the server, dropping back to SSL V3 if TLS V1 cannot be used and SSL V3 is enabled, then dropping back to SSL V2 if SSL V3 cannot be used and SSL V2 is enabled. Note that SSL V2 is not as secure as SSL V3 or TLS V1. This is the prototype for the callback routine provided by the application. It shows the parameters passed to the application callback and the value returned by the callback and should be disabled whenever possible to avoid attacks which force the client and server to drop back to SSL V2 even though they are capable of using SSL V3 or TLS V1.

The server certificate must support key encipherment. This means that the public/private key algorithm must use RSA encryption (the Digital Signature Standard (DSS) does not support data encryption) and the certificate key usage extension (if any) must allow key encipherment.

## **gsk\_secure\_socket\_init()**

The client certificate must support digital signatures. This means the certificate key usage extension (if any) must allow digital signature. The key algorithm can be either the RSA encryption algorithm or the Digital Signature Standard algorithm (DSA).

The SSL server always provides its certificate to the SSL client as part of the handshake. Depending upon the server handshake type, the server may ask the client to provide its certificate. The key label stored in the connection is used to retrieve the certificate from the key database or key ring. The default key will be used if no label is set. The key record must contain both an X.509 certificate and a private key. Refer to the **gsk\_validate\_certificate()** routine for a description of the steps which are performed during certificate validation.

The **gsk\_secure\_socket\_init()** routine can return GSK\_WOULD\_BLOCK\_READ or GSK\_WOULD\_BLOCK\_WRITE if the socket is in non-blocking mode. The connection is not initialized in this case and the application must call **gsk\_secure\_socket\_init()** again when the socket is ready to accept a read request (GSK\_WOULD\_BLOCK\_READ) or a write request (GSK\_WOULD\_BLOCK\_WRITE). The application must provide its own callback routine for **io\_setsocketoptions()** in order to have the SSL handshake processed in non-blocking mode (the default **io\_setsocketoptions()** routine will place the socket into blocking mode during the handshake processing).

## **Related Topics**

**gsk\_environment\_init()**

**gsk\_secure\_socket\_write()**

**gsk\_secure\_socket\_read()**

**gsk\_secure\_socket\_misc()**

**gsk\_secure\_socket\_close()**

## gsk\_secure\_socket\_misc()

Performs miscellaneous secure connection functions.

### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_misc (
    gsk_handle      soc_handle,
    GSK_MISC_ID    misc_id)
```

### Parameters

*soc\_handle*

Specifies the connection handle returned by the **gsk\_secure\_socket\_open()** routine.

*misc\_id*

Miscellaneous function identifier.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ERR\_CONNECTION\_CLOSED]

A close notification alert has been sent for the connection.

#### [GSK\_ERR\_IO]

I/O error communicating with peer application.

#### [GSK\_ERR\_NOT\_SSLV3]

The session is not using the SSL V3 or TLS V1 protocol.

#### [GSK\_ERR\_SOCKET\_CLOSED]

Socket connection closed by peer application.

#### [GSK\_INVALID\_HANDLE]

The connection handle is not valid.

#### GSK\_INVALID\_STATE

The connection is not in the initialized state.

#### [GSK\_MISC\_INVALID\_ID]

The miscellaneous identifier is not valid.

### Usage

The **gsk\_secure\_socket\_misc()** routine performs miscellaneous function for an initialized secure connection.

The following miscellaneous functions are provided:

#### GSK\_RESET\_CIPHER

This function generates new session keys for the connection. A full SSL handshake will be performed if the session has expired or has been reset by the **GSK\_RESET\_SESSION** function. Otherwise a short SSL handshake will be performed. The **GSK\_RESET\_CIPHER** function can be performed only for a session using the SSL V3 or TLS V1 protocol. The **GSK\_RESET\_CIPHER** function initiates the SSL handshake but does not wait for it to complete. Any pending handshake messages will be processed when the **gsk\_secure\_socket\_read()** routine is called to process incoming data.

## **gsk\_secure\_socket\_misc()**

### **GSK\_RESET\_SESSION**

This function resets the session associated with the connection. A full SSL handshake will be performed for the next connection using the session. The current connection is not affected unless the GSK\_RESET\_CIPHER function is performed after the GSK\_RESET\_SESSION function has completed.

## **Related Topics**

**gsk\_secure\_socket\_open()**

**gsk\_secure\_socket\_read()**

**gsk\_secure\_socket\_write()**

## gsk\_secure\_socket\_open()

Creates a secure socket connection.

### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_open (
    gsk_handle      env_handle,
    gsk_handle *    soc_handle)
```

### Parameters

*env\_handle*

Specifies the SSL environment handle returned by the **gsk\_environment\_open()** routine.

*soc\_handle*

Returns the handle for the secure connection. The application should call the **gsk\_secure\_socket\_close()** routine to release the connection when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_INSUFFICIENT\_STORAGE]

Insufficient storage is available.

#### [GSK\_INVALID\_HANDLE]

The environment handle is not valid.

#### [GSK\_INVALID\_STATE]

The environment is not in the initialized state.

### Usage

The **gsk\_secure\_socket\_open()** routine creates a secure socket connection. The connection will be initialized with values obtained from the SSL environment. These values can be changed by the application using the appropriate **gsk\_attribute\_set\_\***() routines. The **gsk\_secure\_socket\_init()** routine should then be called to initialize the connection. This connection can then be used to send and receive data with the remote partner.

### Related Topics

**gsk\_secure\_socket\_close()**

**gsk\_secure\_socket\_init()**

## `gsk_secure_socket_read()`

---

### `gsk_secure_socket_read()`

Reads data using a secure socket connection.

#### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_read (
    gsk_handle    soc_handle,
    char *        buffer,
    int           size,
    int *         length)
```

#### Parameters

*soc\_handle*

Specifies the connection handle returned by the `gsk_secure_socket_open()` routine.

*buffer*

Specifies the buffer to receive the data read from the secure socket connection. The maximum amount of data returned by `gsk_secure_socket_read()` is 16384 (16K) bytes minus the length of the SSL protocol headers.

*size*

Specifies the size of the supplied buffer.

*length*

Returns the length of the data read into the supplied buffer.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

##### **[GSK\_CONNECTION\_ACTIVE]**

A read request is already active for the connection.

##### **[GSK\_ERR\_BAD\_MAC]**

Message verification failed.

##### **[GSK\_ERR\_BAD\_MESSAGE]**

Incorrectly-formatted message received from peer application.

##### **[GSK\_ERR\_BAD\_PEER]**

Peer application has violated the SSL protocol.

##### **[GSK\_ERR\_CONNECTION\_CLOSED]**

Close notification received from peer application.

##### **[GSK\_ERR\_CRYPTO]**

Cryptographic error detected.

##### **[GSK\_ERR\_IO]**

I/O error communicating with peer application.

##### **[GSK\_ERR\_SOCKET\_CLOSED]**

Socket connection closed by peer application.

##### **[GSK\_INSUFFICIENT\_STORAGE]**

Insufficient storage is available.

##### **[GSK\_INVALID\_BUFFER\_SIZE]**

The buffer address or buffer size is not valid.

##### **[GSK\_INVALID\_HANDLE]**

The connection handle is not valid.

**[GSK\_INVALID\_STATE]**

The connection is not in the initialized state.

**[GSK\_WOULD\_BLOCK]**

A complete SSL record is not available.

**[GSK\_WOULD\_BLOCK\_WRITE]**

An SSL handshake is in progress but data cannot be written to the socket.

**Usage**

The **gsk\_secure\_socket\_read()** routine reads data from a secure socket connection and returns it in the application buffer. SSL is a record-based protocol and a single call will never return more than a single SSL record (a maximum of 16384 bytes minus the length of the SSL protocol headers). The application can read an entire SSL record in a single call by supplying a buffer large enough to contain the record. Otherwise, multiple calls will be required to retrieve the entire SSL record.

SSL supports multiple threads but only one thread at a time can call the **gsk\_secure\_socket\_read()** routine for a given connection handle. Multiple concurrent threads can call **gsk\_secure\_socket\_read()** as long as each thread has its own connection handle.

SSL supports sockets in blocking mode and in non-blocking mode. When a socket is in non-blocking mode and a complete SSL record is not available, **gsk\_secure\_socket\_read()** will return with **GSK\_WOULD\_BLOCK**. No data will be returned in the application buffer when **GSK\_WOULD\_BLOCK** is returned. The application should call **gsk\_secure\_socket\_read()** again when there is data available to be read from the socket.

The peer application can initiate an SSL handshake sequence after the connection is established. If this is done and the socket is in non-blocking mode, it is possible for **gsk\_secure\_socket\_read()** to return with **GSK\_WOULD\_BLOCK\_WRITE**. This indicates that an SSL handshake is in progress and the application should call **gsk\_secure\_socket\_read()** again when data can be written to the socket. No data will be returned in the application buffer when **GSK\_WOULD\_BLOCK\_WRITE** is returned.

The application should not read data directly from the socket since this can cause SSL protocol errors if the application inadvertently reads part of an SSL record. If the application must read data from the socket, it is responsible for synchronizing this activity with the peer application so that no SSL records are sent while the application is performing its own read operations.

**Related Topics**

**gsk\_secure\_socket\_write()**

**gsk\_secure\_socket\_init()**

`gsk_secure_socket_shutdown()`

---

## `gsk_secure_socket_shutdown()`

Shuts down a secure socket connection.

### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_shutdown (
    gsk_handle    soc_handle)
```

### Parameters

*soc\_handle*

Specifies the connection handle returned by the `gsk_secure_socket_open()` routine.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

#### **[GSK\_CONNECTION\_ACTIVE]**

The connection has an active write request.

#### **[GSK\_ERR\_CONNECTION\_CLOSED]**

The close notification alert has already been sent.

#### **[GSK\_ERR\_IO]**

I/O error communicating with peer application.

#### **[GSK\_ERR\_NOT\_SSLV3]**

The session is not using the SSL V3 or TLS V1 protocol.

#### **[GSK\_ERR\_SOCKET\_CLOSED]**

Socket connection closed by peer application.

#### **[GSK\_INVALID\_HANDLE]**

The connection handle is not valid.

#### **[GSK\_INVALID\_STATE]**

The connection is not in the initialized state.

#### **[GSK\_WOULD\_BLOCK\_WRITE]**

An attempt to write pending data failed with EWOULDBLOCK.

### Usage

The `gsk_secure_socket_shutdown()` routine will send a close notification alert to the peer application.

Any subsequent calls to the `gsk_secure_socket_write()` routine will return

`GSK_ERR_CONNECTION_CLOSED`. The `gsk_secure_socket_shutdown()` routine cannot be used with the SSL V2 protocol.

The application should call `gsk_secure_socket_shutdown()` before calling `gsk_secure_socket_close()` in order to comply with the SSL V3 and TLS V1 specifications, which require that a close notification alert be sent before closing the transport connection.

For a 1-step shutdown, the application should call the `gsk_secure_socket_shutdown()` routine and then call the `gsk_secure_socket_close()` routine. This sends the close notification alert and then closes the secure socket connection. The application does not wait for acknowledgement from the peer application to the close notification.

## **gsk\_secure\_socket\_shutdown()**

| For a 2-step shutdown, the application should call the **gsk\_secure\_socket\_shutdown()** routine to send the close notification alert and then call the **gsk\_secure\_socket\_read()** routine to process any pending data sent by the peer application. The SSL runtime on the peer system will send a close notification alert when it receives the close notification alert from the local system. The **gsk\_secure\_socket\_read()** routine will return GSK\_ERR\_CONNECTION\_CLOSED when it receives this close notification. The application should then call the **gsk\_secure\_socket\_close()** routine to close the secure socket connection.

## `gsk_secure_socket_write()`

---

### `gsk_secure_socket_write()`

Writes data using a secure socket connection.

#### Format

```
#include <gskssl.h>

gsk_status gsk_secure_socket_write (
    gsk_handle    soc_handle,
    char *        buffer,
    int           size,
    int *         length)
```

#### Parameters

*soc\_handle*

Specifies the connection handle returned by the `gsk_secure_socket_open()` routine.

*buffer*

Specifies the buffer containing the data to write to the secure socket connection.

*size*

Specifies the amount to write.

*length*

Returns the length of the data written.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

##### [GSK\_CONNECTION\_ACTIVE]

A write request is already active for the connection.

##### [GSK\_ERR\_CONNECTION\_CLOSED]

A close notification alert has been sent for the connection.

##### [GSK\_ERR\_CRYPTO]

Cryptographic error detected.

##### [GSK\_ERR\_IO]

I/O error communicating with peer application.

##### [GSK\_ERR\_SOCKET\_CLOSED]

Socket connection closed by peer application.

##### [GSK\_INSUFFICIENT\_STORAGE]

Insufficient storage is available.

##### [GSK\_INVALID\_BUFFER\_SIZE]

The buffer address or buffer size is not valid.

##### [GSK\_INVALID\_HANDLE]

The connection handle is not valid.

##### [GSK\_INVALID\_STATE]

The connection is not in the initialized state.

##### [GSK\_WOULD\_BLOCK]

The SSL record cannot be written to the socket due to an EWOULDBLOCK condition.

#### Usage

The `gsk_secure_socket_write()` routine writes data to a secure socket connection. SSL is a record-based protocol with a maximum record length of 16384 bytes minus the length of the SSL protocol headers.

## **gsk\_secure\_socket\_write()**

| Application data larger than the size of an SSL record will be sent using multiple records. Since **gsk\_secure\_socket\_read()** never returns more than a single SSL record, the receiving application will need to call **gsk\_secure\_socket\_read()** multiple times in order to receive all of the application data when multiple records are needed.

| SSL supports multiple threads but only one thread at a time can call the **gsk\_secure\_socket\_write()** routine for a given connection handle. Multiple concurrent threads can call **gsk\_secure\_socket\_write()** as long as each thread has its own connection handle.

| SSL supports sockets in blocking mode and in non-blocking mode. When a socket is in non-blocking mode and the SSL record cannot be written to the socket, **gsk\_secure\_socket\_write()** will return with **GSK\_WOULD\_BLOCK**. The application must call **gsk\_secure\_socket\_write()** again when the socket is ready to accept more data, specifying the same buffer address and buffer size as the original request. A new write request must not be initiated until the pending write request has been completed as indicated by a return value of 0.

| The application should not write data directly to the socket since this can cause SSL protocol errors if the application inadvertently intermixes its data with SSL protocol data. If the application must write data to the socket, it is responsible for synchronizing this activity with the peer application so that application data is not intermixed with SSL data.

### **Related Topics**

| **gsk\_secure\_socket\_read()**

| **gsk\_secure\_socket\_init()**

## **gsk\_strerror()**

---

### **gsk\_strerror()**

Return a text string for an SSL error code

### **Format**

```
#include <gskssl.h>

const char * gsk_strerror (
    gsk_status      error_code)
```

### **Parameters**

*error\_code*

Specifies an error code returned by a Secure Sockets layer (SSL) routine or by a Certificate Management Services (CMS) routine.

### **Results**

The function return value is the address of the text string. The return value will always be a valid text string address even when the error code is not recognized (the return value will be the string "N/A" in this case).

### **Usage**

The **gsk\_strerror()** routine returns a text string describing an error code returned by an SSL (Secure Sockets layer) or CMS (Certificate Management Services) routine. The **gsk\_strerror()** routine cannot be used to return a text string for an error code returned by one of the deprecated SSL routines. The text string must not be modified or released by the application program.

---

## Chapter 8. Certificate Management Services (CMS) API Reference

This chapter describes the Certificate Management Services (CMS) APIs. These APIs can be used to create/manage your own key database files in a similar function to the SSL gskkyman utility, use certificates stored in the key database file or key ring for purposes other than SSL, and basic PKCS #7 message support.

Following is a list of the Certificate Management Services (CMS) APIs:

- **gsk\_add\_record()** (see page 82)
- **gsk\_change\_database\_password()** (see page 84)
- **gsk\_change\_database\_record\_length()** (see page 86)
- **gsk\_close\_database()** (see page 87)
- **gsk\_close\_directory()** (see page 88)
- **gsk\_copy\_buffer()** (see page 89)
- **gsk\_copy\_certificate()** (see page 90)
- **gsk\_copy\_certificate\_extension()** (see page 91)
- **gsk\_copy\_certification\_request()** (see page 92)
- **gsk\_copy\_content\_info()** (see page 93)
- **gsk\_copy\_crl()** (see page 94)
- **gsk\_copy\_name()** (see page 95)
- **gsk\_copy\_private\_key\_info()** (see page 96)
- **gsk\_copy\_public\_key\_info()** (see page 97)
- **gsk\_copy\_record()** (see page 98)
- **gsk\_create\_certification\_request()** (see page 99)
- **gsk\_create\_database()** (see page 101)
- **gsk\_create\_self\_signed\_certificate()** (see page 103)
- **gsk\_create\_signed\_certificate()** (see page 106)
- **gsk\_create\_signed\_crl()** (see page 108)
- **gsk\_decode\_base64()** (see page 110)
- **gsk\_decode\_certificate()** (see page 111)
- **gsk\_decode\_certificate\_extension()** (see page 112)
- **gsk\_decode\_certification\_request()** (see page 114)
- **gsk\_decode\_crl()** (see page 115)
- **gsk\_decode\_name()** (see page 116)
- **gsk\_delete\_record()** (see page 117)
- **gsk\_dn\_to\_name()** (see page 118)
- **gsk\_encode\_base64()** (see page 120)
- **gsk\_encode\_certificate\_extension()** (see page 121)
- **gsk\_encode\_name()** (see page 123)
- **gsk\_encode\_signature()** (see page 124)
- **gsk\_export\_certificate()** (see page 125)
- **gsk\_export\_certification\_request()** (see page 127)
- **gsk\_export\_key()** (see page 129)
- **gsk\_free\_buffer()** (see page 131)

- | • **gsk\_free\_certificate()** (see page 132)
- | • **gsk\_free\_certificates()** (see page 133)
- | • **gsk\_free\_certificate\_extension()** (see page 134)
- | • **gsk\_free\_certification\_request()** (see page 135)
- | • **gsk\_free\_content\_info()** (see page 136)
- | • **gsk\_free\_crl()** (see page 137)
- | • **gsk\_free\_crls()** (see page 138)
- | • **gsk\_free\_decoded\_extension()** (see page 139)
- | • **gsk\_free\_name()** (see page 140)
- | • **gsk\_free\_private\_key\_info()** (see page 141)
- | • **gsk\_free\_public\_key\_info()** (see page 142)
- | • **gsk\_free\_record()** (see page 143)
- | • **gsk\_free\_records()** (see page 144)
- | • **gsk\_free\_string()** (see page 145)
- | • **gsk\_free\_strings()** (see page 146)
- | • **gsk\_generate\_random\_bytes()** (see page 147)
- | • **gsk\_get\_cms\_vector()** (see page 148)
- | • **gsk\_get\_default\_key()** (see page 149)
- | • **gsk\_get\_default\_label()** (see page 150)
- | • **gsk\_get\_directory\_certificates()** (see page 151)
- | • **gsk\_get\_directory\_crls()** (see page 152)
- | • **gsk\_get\_record\_by\_id()** (see page 154)
- | • **gsk\_get\_record\_by\_index()** (see page 155)
- | • **gsk\_get\_record\_by\_label()** (see page 156)
- | • **gsk\_get\_record\_by\_subject()** (see page 157)
- | • **gsk\_get\_record\_labels()** (see page 158)
- | • **gsk\_get\_update\_code()** (see page 159)
- | • **gsk\_import\_certificate()** (see page 160)
- | • **gsk\_import\_key()** (see page 162)
- | • **gsk\_make\_content\_msg()** (see page 164)
- | • **gsk\_make\_data\_content()** (see page 165)
- | • **gsk\_make\_data\_msg()** (see page 166)
- | • **gsk\_make\_encrypted\_data\_content()** (see page 167)
- | • **gsk\_make\_encrypted\_data\_msg()** (see page 169)
- | • **gsk\_make\_enveloped\_data\_content()** (see page 171)
- | • **gsk\_make\_enveloped\_data\_msg()** (see page 173)
- | • **gsk\_make\_signed\_data\_content()** (see page 175)
- | • **gsk\_make\_signed\_data\_msg()** (see page 177)
- | • **gsk\_make\_wrapped\_content()** (see page 179)
- | • **gsk\_mktime()** (see page 180)
- | • **gsk\_name\_compare()** (see page 181)
- | • **gsk\_name\_to\_dn()**(see page 182)
- | • **gsk\_open\_database()** (see page 183)
- | • **gsk\_open\_database\_using\_stash\_file()** (see page 185)
- | • **gsk\_open\_directory()** (see page 187)

- | • **gsk\_open\_keyring()** (see page 188)
- | • **gsk\_query\_crypto\_level()** (see page 189)
- | • **gsk\_query\_database\_label()** (see page 190)
- | • **gsk\_query\_database\_record\_length()** (see page 191)
- | • **gsk\_rdttime()** (see page 192)
- | • **gsk\_read\_content\_msg()** (see page 193)
- | • **gsk\_read\_data\_content()** (see page 194)
- | • **gsk\_read\_data\_msg()** (see page 195)
- | • **gsk\_read\_encrypted\_data\_content()** (see page 196)
- | • **gsk\_read\_encrypted\_data\_msg()** (see page 198)
- | • **gsk\_read\_enveloped\_data\_content()** (see page 200)
- | • **gsk\_read\_enveloped\_data\_msg()** (see page 202)
- | • **gsk\_read\_signed\_data\_content()** (see page 204)
- | • **gsk\_read\_signed\_data\_msg()** (see page 206)
- | • **gsk\_read\_wrapped\_content()** (see page 208)
- | • **gsk\_receive\_certificate()** (see page 209)
- | • **gsk\_replace\_record()** (see page 210)
- | • **gsk\_set\_default\_key()** (see page 212)
- | • **gsk\_sign\_certificate()** (see page 213)
- | • **gsk\_sign\_crl()** (see page 214)
- | • **gsk\_sign\_data()** (see page 215)
- | • **gsk\_validate\_certificate()** (see page 217)
- | • **gsk\_verify\_certificate\_signature()** (see page 220)
- | • **gsk\_verify\_crl\_signature()** (see page 221)
- | • **gsk\_verify\_data\_signature()** (see page 222)

## **gsk\_add\_record()**

---

### **gsk\_add\_record()**

Adds a record to a key or request database.

### **Format**

```
#include <gskcms.h>
gsk_status gsk_add_record (
    gsk_handle      db_handle,
    gskdb_record *  record)
```

### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine.

*record*

Specifies the database record.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

#### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

#### **[CMSERR\_DUPLICATE\_CERTIFICATE]**

The database already contains the certificate.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The record type is not supported for the database type.

#### **[CMSERR\_IO\_ERROR]**

Unable to write record.

#### **[CMSERR\_LABEL\_NOT\_UNIQUE]**

The record label is not unique.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **[CMSERR\_NO\_PRIVATE\_KEY]**

No private key is provided for a record type that requires a private key.

#### **[CMSERR\_RECORD\_TOO\_BIG]**

The record is larger than the database record length.

#### **[CMSERR\_RECTYPE\_NOT\_VALID]**

The record type is not valid.

#### **[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

### **Usage**

The **gsk\_add\_record()** routine adds a record to a key or request database. The database must be open for update in order to add records. Unused and reserved fields in the **gskdb\_record** structure must be

## **gsk\_add\_record()**

initialized to zero. An error will be returned when adding a certificate to a key database if the database already contains the certificate. If the record has a private key, the encrypted private key will be generated from the private key supplied in the database record.

The *recordType* field identifies the database record type as follows:

### **gskdb\_rectype\_certificate**

The record contains a X.509 certificate

### **gskdb\_rectype\_certKey**

The record contains a X.509 certificate and private key

### **gskdb\_rectype\_keyPair**

The record contains a PKCS #10 certification request and private key

The *recordFlags* field is a bit field with the following values:

### **GSKDB\_RECFLAG\_TRUSTED**

The certificate is trusted

### **GSKDB\_RECFLAG\_DEFAULT**

This is the default key

A unique record identifier is assigned when the record is added to the database and will be returned to the application in the *recordId* field. If the record contains a X.509 certificate, the *issuerRecordId* field will be set to the record identifier of the certificate issuer.

The record label is used as a friendly name for the database entry and is in the local code page. It can be set to any value and consists of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be set to an empty string.

If the record contains a certificate, the certificate will be validated and the record will not be added to the database if the validation check fails.

With the exception of the record label, all character strings are specified using UTF-8.

The database file is updated as part of the **gsk\_add\_record()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## `gsk_change_database_password()`

---

### `gsk_change_database_password()`

Changes the database password.

#### Format

```
#include <gskcms.h>

gsk_status gsk_change_database_password (
    const char *   filename,
    const char *   old_password,
    const char *   new_password,
    gsk_time       pwd_expiration)
```

#### Parameters

##### *filename*

Specifies the database file name in the local code page. The length of the fully-qualified file name cannot exceed 251.

##### *old\_password*

Specifies the current database password in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

##### *new\_password*

Specifies the new database password in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

##### *pwd\_expiration*

Specifies the new password expiration time as the number of seconds since the POSIX epoch. A value of 0 indicates the password does not expire.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_ACCESS\_DENIED]**

The file permissions do not allow access.

##### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

##### **[CMSERR\_BAD\_FILENAME]**

The database file name is not valid.

##### **[CMSERR\_DB\_CORRUPTED]**

The database file is not valid.

##### **[CMSERR\_DB\_LOCKED]**

The database is open for update by another process.

##### **[CMSERR\_FILE\_NOT\_FOUND]**

The database file is not found.

##### **[CMSERR\_IO\_CANCELED]**

The user canceled the password prompt.

##### **[CMSERR\_IO\_ERROR]**

An input/output request failed.

##### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

## **gsk\_change\_database\_password()**

### **[CMSERR\_OPEN\_FAILED]**

Unable to open the database.

### **Usage**

The **gsk\_change\_database\_password()** routine will change the password for the database and set a new password expiration time. **gsk\_mktime()** can be used to convert a year/month/day time value to the number of seconds since the POSIX epoch.

## `gsk_change_database_record_length()`

---

### `gsk_change_database_record_length()`

Changes the database record length.

#### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_change_database_record_length (
    gsk_handle      db_handle,
    gsk_size        record_length)
```

#### Parameters

*db\_handle*

Specifies the database handle returned by the `gsk_create_database()` routine or the `gsk_open_database()` routine.

*record\_length*

Specifies the new database record length. The default record length will be used if zero is specified for this parameter. All records in the database will have this length. The minimum record length is 2500.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

##### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_IO\_ERROR]**

An input/output request failed.

##### **[CMSERR\_LENGTH\_TOO\_SMALL]**

The record length is less than the minimum value.

##### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

##### **[CMSERR\_RECORD\_TOO\_BIG]**

A record in the database is larger than the new record length.

##### **[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

#### Usage

The `gsk_change_database_record_length()` routine will change the record length for the database. All records in the database have the same length and a database entry cannot span records. An error will be returned if the requested record length is smaller than the largest entry in the database.

---

## gsk\_close\_database()

Closes a key or request database.

### Format

```
#include <gskcms.h>

gsk_status gsk_close_database (
    gsk_handle *    db_handle)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine. The handle will be set to NULL upon successful completion.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

### Usage

The **gsk\_close\_database()** routine will close a key or request database. The *db\_handle* will not be valid upon return from the **gsk\_close\_database()** routine.

## **gsk\_close\_directory()**

---

### **gsk\_close\_directory()**

Closes an LDAP directory.

### **Format**

```
#include <gskcms.h>

gsk_status gsk_close_directory (
    gsk_handle *    directory_handle)
```

### **Parameters**

*directory\_handle*

Specifies the directory handle returned by the **gsk\_open\_directory()** routine. The handle will be set to NULL upon successful completion.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

#### **[CMSERR\_BAD\_HANDLE]**

The directory handle is not valid.

### **Usage**

The **gsk\_close\_directory()** routine will close an LDAP directory opened by the **gsk\_open\_directory()** routine. The *directory\_handle* will not be valid upon return from the **gsk\_close\_directory()** routine.

## gsk\_copy\_buffer()

Copies a buffer.

### Format

```
#include <gskcms.h>

gsk_status gsk_copy_buffer (
    gsk_buffer *    in_buffer,
    gsk_buffer *    out_buffer)
```

### Parameters

*in\_buffer*

Specifies the source buffer.

*out\_buffer*

Specifies the destination buffer. The application should call the **gsk\_free\_buffer()** routine when the buffer is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **gsk\_copy\_buffer()** routine will allocate the output buffer and then copy the input buffer to the output buffer. Storage for the base **gsk\_buffer** structure is provided by the caller.

## `gsk_copy_certificate()`

---

### `gsk_copy_certificate()`

Copies a X.509 certificate.

### Format

```
#include <gskcms.h>

gsk_status gsk_copy_certificate (
    x509_certificate *   in_certificate,
    x509_certificate *   out_certificate)
```

### Parameters

*in\_certificate*

Specifies the source certificate.

*out\_certificate*

Specifies the destination certificate. The application should call the **`gsk_free_certificate()`** routine when the certificate is no longer needed.

### Results

The function return value will be 0 (**`GSK_OK`**) if no error is detected. Otherwise, it will be one of the return codes listed in the **`gskcms.h`** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **`gsk_copy_certificate()`** routine will allocate the output certificate and then copy the input certificate to the output certificate. Storage for the base `x509_certificate` structure is provided by the caller.

---

## gsk\_copy\_certificate\_extension()

Copies a X.509 certificate extension.

### Format

```
#include <gskcms.h>

gsk_status gsk_copy_certificate_extension (
    x509_extension *    in_extension,
    x509_extension *    out_extension)
```

### Parameters

*in\_extension*

Specifies the source certificate extension.

*out\_extension*

Specifies the destination certificate extension. The application should call the **gsk\_free\_certificate\_extension()** routine when the extension is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **gsk\_copy\_certificate\_extension()** routine will allocate the output certificate extension and then copy the input certificate extension to the output certificate extension. Storage for the base x509\_extension structure is provided by the caller.

## **gsk\_copy\_certification\_request()**

---

### **gsk\_copy\_certification\_request()**

Copies a PKCS #10 certification request.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_copy_certification_request (
    pkcs_cert_request *   in_request,
    pkcs_cert_request *   out_request)
```

### **Parameters**

*in\_request*

Specifies the source certification request.

*out\_request*

Specifies the destination certification request. The application should call the **gsk\_free\_certification\_request()** routine when the certification request is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_copy\_certification\_request()** routine will allocate the output certification request and then copy the input certification request to the output certification request. Storage for the base `pkcs_cert_request` structure is provided by the application.

## gsk\_copy\_content\_info()

Copies PKCS #7 content information.

### Format

```
#include <gskcms.h>

gsk_status gsk_copy_content_info (
    pkcs_content_info *    in_info,
    pkcs_content_info *    out_info)
```

### Parameters

*in\_info*

Specifies the source content information.

*out\_info*

Specifies the destination content information. The application should call the **gsk\_free\_content\_info()** routine when the content information is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **gsk\_copy\_content\_info()** routine will allocate the output content information and then copy the input content information to the output content information. Storage for the base `pkcs_content_info` structure is provided by the application.

## **gsk\_copy\_crl()**

---

### **gsk\_copy\_crl()**

Copies a X.509 certificate revocation list.

### **Format**

```
#include <gskcms.h>

gsk_status gsk_copy_crl (
    x509_crl *    in_crl,
    x509_crl *    out_crl)
```

### **Parameters**

*in\_crl*

Specifies the source certificate revocation list.

*out\_crl*

Specifies the destination certificate revocation list. The application should call the **gsk\_free\_crl()** routine when the certificate revocation list is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_copy\_crl()** routine will allocate the output certificate revocation list and then copy the input list to the output list. Storage for the base x509\_crl structure is provided by the caller.

## gsk\_copy\_name()

Copies a X.509 name.

### Format

```
#include <gskcms.h>

gsk_status gsk_copy_name (
    x509_name *    in_name,
    x509_name *    out_name)
```

### Parameters

*in\_name*

Specifies the source name.

*out\_name*

Specifies the destination name. The application should call the **gsk\_free\_name()** routine when the name is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **gsk\_copy\_name()** routine will allocate the output name and then copy the input name to the output name. Storage for the base x509\_name structure is provided by the caller.

## **gsk\_copy\_private\_key\_info()**

---

### **gsk\_copy\_private\_key\_info()**

Copies the private key information.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_copy_private_key_info (
    pkcs_private_key_info *   in_info,
    pkcs_private_key_info *   out_info)
```

### **Parameters**

*in\_info*

Specifies the source private key information.

*out\_info*

Specifies the destination private key information. The application should call the **gsk\_free\_private\_key\_info()** routine when the private key is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_copy\_private\_key\_info()** routine will allocate the output private key and then copy the input key to the output key. Storage for the base `pkcs_private_key_info` structure is provided by the caller.

## gsk\_copy\_public\_key\_info()

Copies the public key information.

### Format

```
#include <gskcms.h>

gsk_status gsk_copy_public_key_info (
    x509_public_key_info *   in_info,
    x509_public_key_info *   out_info)
```

### Parameters

*in\_info*

Specifies the source public key information.

*out\_info*

Specifies the destination public key information. The application should call the **gsk\_free\_public\_key\_info()** routine when the public key is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **gsk\_copy\_public\_key\_info()** routine will allocate the output public key and then copy the input key to the output key. Storage for the base `x509_public_key_info` structure is provided by the caller.

## **gsk\_copy\_record()**

---

### **gsk\_copy\_record()**

Copies a database record.

### **Format**

```
#include <gskcms.h>

gsk_status gsk_copy_record (
    gskdb_record *    in_record,
    gskdb_record **  out_record)
```

### **Parameters**

*in\_record*

Specifies the source record.

*out\_record*

Returns the copied record. The application should call the **gsk\_free\_record()** routine when the record is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_copy\_record()** routine will allocate the output record and then copy the input record to the output record. The address of the copied record will then be returned to the application.

## gsk\_create\_certification\_request()

Creates a PKCS #10 certification request.

### Format

```
#include <gskcms.h>

gsk_status gsk_create_certification_request (
    gsk_handle          db_handle,
    const char *        label,
    x509_algorithm_type signature_algorithm,
    int                 key_size,
    const char *        subject_name,
    x509_extensions *   extensions)
```

### Parameters

#### *db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine. This must be a request database and not a key database.

#### *label*

Specifies the label for the new database record. The label is specified in the local code page.

#### *signature\_algorithm*

Specifies the signature algorithm for the certificate.

#### *key\_size*

Specifies the key size in bits.

#### *subject\_name*

Specifies the distinguished name for the certificate subject. The distinguished name is specified in the local code page and consists of one or more relative distinguished name components separated by commas.

#### *extensions*

Specifies certificate extensions to be included in the certification request. Specify NULL for this parameter if no certificate extensions are provided.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

#### **[CMSERR\_BAD\_HANDLE ]**

The database handle is not valid.

#### **[CMSERR\_BAD\_KEY\_SIZE]**

The key size is not valid.

#### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certification requests.

#### **[CMSERR\_IO\_ERROR]**

Unable to write record.

## **gsk\_create\_certification\_request()**

### **[CMSERR\_LABEL\_NOT\_UNIQUE]**

The record label is not unique.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_RECORD\_TOO\_BIG]**

The record is larger than the database record length.

### **[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

## **Usage**

The **gsk\_create\_certification\_request()** routine creates a request for a new certificate as described in PKCS #10 (Certification Request Syntax Standard). The request is then stored in the request database.

The **gsk\_export\_certification\_request()** routine can be called to create an export file containing the request for transmission to the certification authority.

The following signature algorithms are supported:

### **x509\_alg\_md2WithRsaEncryption**

RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}

### **x509\_alg\_md5WithRsaEncryption**

RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}

### **x509\_alg\_sha1WithRsaEncryption**

RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}

### **x509\_alg\_dsaWithSha1**

Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}

An RSA key size must be between 512 and 2048 bits and will be rounded up to a multiple of 16 bits. A DSA key size must be between 512 and 1024 bits and will be rounded up to a multiple of 64 bits.

The record label is used as a friendly name for the database entry. It can be any value and consists of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string.

The *extensions* parameter can be used to provide certificate extensions for inclusion in the certification request. Whether or not a particular certificate extension will be included in the new certificate is determined by the certification authority.

The database must be open for update in order to add the new request. The database file is updated as part of the **gsk\_create\_certification\_request()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## gsk\_create\_database()

Creates a key or request database.

### Format

```
#include <gskcms.h>

gsk_status gsk_create_database (
    char *          filename,
    char *          password,
    gskdb_database_type db_type,
    gsk_size        record_length,
    gsk_time        pwd_expiration,
    gsk_handle *    db_handle)
```

### Parameters

#### *filename*

Specifies the database file name in the local code page. The length of the fully-qualified file name cannot exceed 251.

#### *password*

Specifies the database password in the local code page. The password must consist of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string. The user will be prompted to enter the password if NULL is specified for this parameter.

#### *db\_type*

Specifies the database type and must be `gskdb_dbtype_key` for a key database or `gskdb_dbtype_request` for a certification request database.

#### *record\_length*

Specifies the database record length. The default record length will be used if zero is specified for this parameter. All records in the database will have this length. The minimum record length is 2500.

#### *pwd\_expiration*

Specifies the database password expiration time as the number of seconds since the POSIX epoch. A value of 0 indicates the password does not expire.

#### *db\_handle*

Returns the database handle. The application should call the `gsk_close_database()` routine when it no longer needs access to the database.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### **[CMSERR\_BAD\_FILENAME]**

The database file name is not valid.

#### **[CMSERR\_DB\_EXISTS]**

The database already exists.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type is not valid.

#### **[CMSERR\_IO\_CANCELED]**

The user canceled the password prompt.

#### **[CMSERR\_IO\_ERROR]**

An input/output request failed.

## **gsk\_create\_database()**

### **[CMSERR\_LENGTH\_TOO\_SMALL]**

The record length is less than the minimum value.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_OPEN\_FAILED]**

Unable to open the key database.

## **Usage**

The **gsk\_create\_database()** routine will create a key or request database. The database must not already exist. A new key database will contain an initial set of Certificate Authority certificates for use in validating certificate signatures.

## gsk\_create\_self\_signed\_certificate()

Creates a self-signed certificate.

### Format

```
#include <gskcms.h>

gsk_status gsk_create_self_signed_certificate (
    gsk_handle          db_handle,
    const char *       label,
    x509_algorithm_type signature_algorithm,
    int                key_size,
    const char *       subject_name,
    int                num_days,
    gsk_boolean        ca_certificate,
    x509_extensions *  extensions)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine. This must be a key database and not a request database.

*label*

Specifies the label for the new database record. The label is specified in the local code page.

*signature\_algorithm*

Specifies the certificate signature algorithm.

*key\_size*

Specifies the key size in bits.

*subject\_name*

Specifies the distinguished name for the certificate subject. The distinguished name is specified in the local code page and consists of one or more relative distinguished name components separated by commas.

*num\_days*

Specifies the number of days for the certificate validity period as a value between 1 and 9999 (the maximum of 9999 will be used if a larger value is specified).

*ca\_certificate*

Specify TRUE if this is a certification authority certificate or FALSE if this is an end user certificate.

*extensions*

Specifies the certificate extensions for the new certificate. Specify NULL for this parameter if no certificate extensions are supplied.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BACKUP\_EXISTS]

The backup file already exists.

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_BAD\_KEY\_SIZE]

The key size is not valid.

## **gsk\_create\_self\_signed\_certificate()**

### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

### **[CMSERR\_BAD\_SUBJECT\_NAME]**

The subject name is not valid.

### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

### **[CMSERR\_IO\_ERROR]**

Unable to write record.

### **[CMSERR\_LABEL\_NOT\_UNIQUE]**

The record label is not unique.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_RECORD\_TOO\_BIG]**

The record is larger than the database record length.

### **[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

## **Usage**

The **gsk\_create\_self\_signed\_certificate()** routine will generate a self-signed X.509 certificate as described in RFC 2459 (X.509 Public Key Infrastructure). A certification authority certificate will have basic constraints and key usage extensions which allow the certificate to be used to sign other certificates and certificate revocation lists. An end user certificate will have basic constraints and key usage extensions which allow the certificate to be used for authentication, digital signatures, and data encryption (except for a DSA key which cannot be used for data encryption). The new certificate is then stored in the key database. The **gsk\_export\_certificate()** routine can be called to create an export file containing the certificate for transmission to another system.

The following signature algorithms are supported:

### **x509\_alg\_md2WithRsaEncryption**

RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}

### **x509\_alg\_md5WithRsaEncryption**

RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}

### **x509\_alg\_sha1WithRsaEncryption**

RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}

### **x509\_alg\_dsaWithSha1**

Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}

An RSA key size must be between 512 and 2048 bits and will be rounded up to a multiple of 16 bits. A DSA key size must be between 512 and 1024 bits and will be rounded up to a multiple of 64 bits.

The record label is used as a friendly name for the database entry. It can be any value and consists of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string.

A CA certificate will have SubjectKeyIdentifier, KeyUsage and BasicConstraints extensions while an end user certificate will have SubjectKeyIdentifier and KeyUsage extensions. The application can supply additional extensions through the *extensions* parameter. A KeyUsage or BasicConstraints extension provided by the application will replace the default extension created for the certificate. A SubjectKeyIdentifier extension provided by the application will be ignored.

### **gsk\_create\_self\_signed\_certificate()**

| The database must be open for update in order to add the new certificate. The database file is updated as  
| part of the **gsk\_create\_self\_signed\_certificate()** processing. A temporary database file is created using  
| the same name as the database file with ".new" appended to the name. The database file is then  
| overwritten and the temporary database file is deleted. The temporary database file will not be deleted if  
| an error occurs while rewriting the database file.

## **gsk\_create\_signed\_certificate()**

---

### **gsk\_create\_signed\_certificate()**

Creates a signed certificate.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_create_signed_certificate (
    gsk_handle          db_handle,
    const char *        label,
    int                 num_days,
    gsk_boolean         ca_certificate,
    x509_extensions *  extensions,
    gsk_buffer *        cert_request,
    gsk_buffer *        signed_certificate)
```

#### **Parameters**

##### *db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine. This must be a key database and not a request database.

##### *label*

Specifies the label for the certificate to be used to sign the new certificate. The label is specified in the local code page.

##### *num\_days*

Specifies the number of days for the certificate validity period as a value between 1 and 9999 (the maximum of 9999 will be used if a larger value is specified).

##### *ca\_certificate*

Specify TRUE if this is a certification authority certificate or FALSE if this is an end user certificate.

##### *extensions*

Specifies the certificate extensions for the new certificate. Specify NULL for this parameter if no certificate extensions are supplied.

##### *cert\_request*

Specifies the PKCS #10 certification request stream in either binary DER-encoded format or in Base64 format. A Base64 stream is in the local code page.

##### *signed\_certificate*

Returns the signed certificate in Base64 format. The Base64 stream will be in the local code page. The application should call the **gsk\_free\_buffer()** routine to release the certificate stream when it is no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_BAD\_ENCODING]**

The certificate request stream is not valid.

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

##### **[CMSERR\_BAD\_SIGNATURE]**

The request signature is not correct.

**[CMSERR\_EXPIRED]**

The signer certificate is expired.

**[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

**[CMSERR\_INCORRECT\_KEY\_USAGE]**

The signer certificate key usage does not allow signing certificates.

**[CMSERR\_ISSUER\_NOT\_CA]**

The signer certificate is not for a certification authority.

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

**[CMSERR\_NO\_PRIVATE\_KEY]**

The signer certificate does not have a private key.

**[CMSERR\_RECORD\_NOT\_FOUND]**

The signer certificate is not found in the key database.

**[CMSERR\_SUBJECT\_IS\_CA]**

The requested subject name is the same as the signer name.

## Usage

The **gsk\_create\_signed\_certificate()** routine will generate a X.509 certificate as described in RFC 2459 (X.509 Public Key Infrastructure). The new certificate will be signed using the certificate specified by the *label* parameter. A certification authority certificate will have basic constraints and key usage extensions which allow the certificate to be used to sign other certificates and certificate revocation lists. An end user certificate will have basic constraints and key usage extensions which allow the certificate to be used for authentication, digital signatures, and data encryption (except for a DSA key which cannot be used for data encryption). The certificate expiration date will be set to the earlier of the requested expiration date and the expiration date of the signing certificate.

The signing certificate must have an associated private key, the BasicConstraints extension must either be omitted or must have the CA indicator set, and the KeyUsage extension must either be omitted or must allow signing certificates. The subject key identifier from the signing certification will be copied to the signed certificate as the authority key identifier.

A CA certificate will have SubjectKeyIdentifier, KeyUsage and BasicConstraints extensions while an end user certificate will have SubjectKeyIdentifier and KeyUsage extensions. An end user certificate will also have an AuthorityKeyIdentifier extension if the signing certificate has a SubjectKeyIdentifier extension. The application can supply additional extensions through the *extensions* parameter. An AuthorityKeyIdentifier, KeyUsage or BasicConstraints extension provided by the application will replace the default extension created for the certificate. A SubjectKeyIdentifier extension provided by the application will be ignored.

Certificate extensions can also be contained within the certification request. A certificate extension supplied by the application will override a certificate extension of the same type contained in the certification request. The certificate extensions found in the certification request will be copied unmodified to the new certificate with the following exceptions:

- The AuthorityInfoAccess, AuthorityKeyIdentifier, BasicConstraints, CrlDistributionPoints, IssuerAltName, NameConstraints, PolicyConstraints, PolicyMappings, and PrivateKeyUsagePeriod extensions will not be copied
- The keyCertSign and crlSign flags in the KeyUsage extension will be modified based upon the value of the *ca\_certificate* parameter.

No certification path validation is performed by the **gsk\_create\_signed\_certificate()** routine. An error will be returned if the requested subject name is the same as the subject name in the signing certificate.

## **gsk\_create\_signed\_crl()**

---

### **gsk\_create\_signed\_crl()**

Creates a signed certificate revocation list.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_create_signed_crl (
    gsk_handle          db_handle,
    const char *       label,
    gsk_int32          crl_number,
    int                num_days,
    x509_revoked_certificates *
    x509_extensions *  revoked_certificates,
    gsk_buffer *       extensions,
    gsk_buffer *       signed_crl)
```

#### **Parameters**

##### *db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine. This must be a key database and not a request database.

##### *label*

Specifies the label for the certificate to be used to sign the certificate revocation list. The label is specified in the local code page.

##### *crl\_number*

Specifies the CRL number. Each CRL is numbered with each successive revocation list having a larger CRL number than all previous revocation lists.

##### *num\_days*

Specifies the number of days until the next CRL will be issued and is specified as a value between 1 and 9999 (the maximum of 9999 will be used if a larger value is specified).

##### *revoked\_certificates*

Specifies the revoked certificates to be included in the CRL.

##### *extensions*

Specifies the CRL extensions for the new CRL. Specify NULL for this parameter if no CRL extensions are supplied.

##### *signed\_crl*

Returns the signed certificate revocation list in Base64 format. The Base64 stream will be in the local code page. The application should call the **gsk\_free\_buffer()** routine to release the stream when it is no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

##### **[CMSERR\_BAD\_SIGNATURE]**

The request signature is not correct.

##### **[CMSERR\_EXPIRED]**

The signer certificate is expired.

**[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

**[CMSERR\_INCORRECT\_KEY\_USAGE]**

The signer certificate key usage does not allow signing a CRL.

**[CMSERR\_ISSUER\_NOT\_CA]**

The signer certificate is not for a certification authority.

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

**[CMSERR\_NO\_PRIVATE\_KEY]**

The signer certificate does not have a private key.

**[CMSERR\_RECORD\_NOT\_FOUND]**

The signer certificate is not found in the key database.

## Usage

The **gsk\_create\_signed\_crl()** routine will generate a X.509 certificate revocation list (CRL) as described in RFC 2459 (X.509 Public Key Infrastructure). The new CRL will be signed using the certificate specified by the *label* parameter. The number of days until the next CRL is issued will be set to the earlier of the requested date and the expiration of the signing certificate.

The signing certificate must have an associated private key, the BasicConstraints extension must either be omitted or must have the CA indicator set, and the KeyUsage extension must either be omitted or must allow signing certificate revocation lists. The subject key identifier from the signing certification will be copied to the signed CRL as the authority key identifier.

The CRL will have a CRLNumber extension containing the value specified by the *crl\_number* parameter. It will also have an AuthorityKeyIdentifier extension if the signing certificate has a SubjectKeyIdentifier extension. The application can supply additional extensions through the *extensions* parameter. An AuthorityKeyIdentifier or CRLNumber extension provided by the application will replace the default extension created for the CRL.

No certification path validation is performed by the **gsk\_create\_signed\_crl()** routine.

## **gsk\_decode\_base64()**

---

### **gsk\_decode\_base64()**

Decodes a Base64-encoded stream.

### **Format**

```
#include <gskcms.h>

gsk_status gsk_decode_base64 (
    gsk_buffer *    encoded_stream,
    gsk_buffer *    decoded_stream)
```

### **Parameters**

*encoded\_stream*

Specifies the Base64-encoded stream. The encoded data must be in the local code page.

*decoded\_stream*

Returns the decoded stream. The application should call the **gsk\_free\_buffer()** routine to release the decoded stream when it is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BAD\_BASE64\_ENCODING]**

Incorrect Base64 encoding.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_decode\_base64()** routine will decode a Base64-encoded stream created by the **gsk\_encode\_base64()** routine. The encoded stream must be in the local code page and must not include any header or trailer lines added by the application to identify the stream contents (such as '-----BEGIN CERTIFICATE-----' or '-----END CERTIFICATE-----'). New line characters and whitespace characters (tabs and spaces) are ignored.

## gsk\_decode\_certificate()

Decodes a X.509 certificate.

### Format

```
#include <gskcms.h>

gsk_status gsk_decode_certificate (
    gsk_buffer *      stream,
    x509_certificate * certificate)
```

### Parameters

*stream*

Specifies the encoded certificate.

*certificate*

Returns the decoded certificate information. The application should call the **gsk\_free\_certificate()** routine to release the decoded certificate when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

#### [ASN\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_decode\_certificate()** routine decodes a X.509 certificate and returns the decoded information to the application. The certificate must have been encoded as described in RFC 2459 (Internet X.509 Public Key Infrastructure). The *derCertificate* field will contain the undecoded TBSCertificate ASN.1 sequence for use in verifying the certificate signature, the *tbsCertificate* field will contain the decoded TBSCertificate ASN.1 sequence, and the *signatureAlgorithm* and *signatureValue* fields will contain the certificate signature. The **gsk\_encode\_signature()** routine can be used to recreate the encoded certificate from the *x509\_certificate* structure returned by the **gsk\_decode\_certificate()** routine.

Character strings contained in the certificate will be returned using UTF-8 encoding. The application can call **iconv()** to convert the string to a different encoding as needed.

The certificate extensions will be returned with the extension values in ASN.1 encoded format. The **gsk\_decode\_certificate\_extension()** routine can be called to decode a particular certificate extension. This allows all of the certificate extensions to be returned even when one or more extensions cannot be processed by the System SSL runtime.

## **gsk\_decode\_certificate\_extension()**

---

### **gsk\_decode\_certificate\_extension()**

Decodes a X.509 certificate extension.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_decode_certificate_extension (
    x509_extension *      encoded_extension,
    x509_decoded_extension *  decoded_extension)
```

#### **Parameters**

*encoded\_extension*

Specifies the encoded X.509 extension as returned by the **gsk\_decode\_certificate()** or **gsk\_decode\_crl()** routine.

*decoded\_extension*

Returns the decoded extension data. The application should call the **gsk\_free\_decoded\_extension()** routine to release the decoded extension when it is no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[ASN\_NO\_MEMORY]**

Insufficient memory is available.

##### **[CMSERR\_EXT\_NOT\_SUPPORTED]**

The certificate extension is not supported.

##### **[CMSERR\_NO\_MEMORY]**

Insufficient memory is available.

#### **Usage**

The **gsk\_decode\_certificate()** and **gsk\_decode\_crl()** routines returns all of the certificate extensions in the `x509_extensions` structure with the extension values still in ASN.1 encoded format. The application then calls the **gsk\_decode\_certificate\_extension()** routine to decode a specific certificate extension.

The **gsk\_decode\_certificate\_extension()** routine returns character strings using UTF-8 encoding. If necessary, the application can call the **iconv()** routine to convert the strings to a different encoding.

The following certificate extensions are supported:

- AuthorityInfoAccess
- AuthorityKeyIdentifier
- BasicConstraints
- CertificateIssuer
- CertificatePolicies
- CrlDistributionPoints
- CrlNumber
- CrlReasonCode
- DeltaCrlIndicator
- ExtKeyUsage
- HoldInstructionCode

## `gsk_decode_certificate_extension()`

- | • InvalidityDate
- | • IssuerAltName
- | • IssuingDistributionPoint
- | • KeyUsage
- | • NameConstraints
- | • PolicyConstraints
- | • PolicyMappings
- | • PrivateKeyUsagePeriod
- | • SubjectAltName
- | • SubjectDirectoryAttributes
- | • SubjectKeyIdentifier

| The following general name types are supported:

- | • DirectoryName
- | • DnsName
- | • IpAddress
- | • RegisteredId
- | • Rfc822Name
- | • UniformResourceIdentifier

| Refer to RFC 2459 (Internet X.509 Public Key Infrastructure) for more information about the various certificate extensions.

## `gsk_decode_certification_request()`

---

### `gsk_decode_certification_request()`

Decodes a PKCS #10 certification request.

#### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_decode_certification_request (
    gsk_buffer *      stream,
    pkcs_cert_request * request)
```

#### Parameters

*stream*

Specifies the encoded certification request.

*request*

Returns the decoded certification request. The application should call the `gsk_free_certification_request()` routine to release the decoded certification request when it is no longer needed.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following is a common error:

##### [ASN\_NO\_MEMORY]

Insufficient memory is available.

#### Usage

The `gsk_decode_certification_request()` routine decodes a Public Key Cryptography Standards (PKCS) certification request and returns the decoded information to the application. The request must have been encoded as described in PKCS #10 (Certification Request Syntax Standard). The *derRequestInfo* field will contain the undecoded CertificationRequestInfo ASN.1 sequence for use in verifying the request signature, the *certificationRequestInfo* field will contain the decoded CertificationRequestInfo ASN.1 sequence, and the *signatureAlgorithm* and *signatureValue* fields will contain the request signature. The `gsk_encode_signature()` routine can be used to recreate the encoded certification request from the `pkcs_cert_request` structure returned by the `gsk_decode_certification_request()` routine.

Character strings contained in the request will be returned using UTF-8 encoding. If necessary, the application can call `iconv()` to convert the string to a different encoding.

## gsk\_decode\_crl()

Decodes a X.509 certificate revocation list.

### Format

```
#include <gskcms.h>

gsk_status gsk_decode_crl (
    gsk_buffer *    stream,
    x509_crl *     crl)
```

### Parameters

*stream*

Specifies the encoded certificate revocation list.

*crl* Returns the decoded information. The application should call the **gsk\_free\_crl()** routine to release the decoded certificate revocation list when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

#### [ASN\_NO\_MEMORY]

Insufficient memory is available.

### Usage

The **gsk\_decode\_crl()** routine decodes a X.509 certificate revocation list (CRL) and returns the decoded information to the application. The CRL must have been encoded as described in RFC 2459 (Internet X.509 Public Key Infrastructure). The *derCertList* field will contain the undecoded TBSCertList ASN.1 sequence for use in verifying the certificate signature, the *tbsCertList* field will contain the decoded TBSCertList ASN.1 sequence, and the *signatureAlgorithm* and *signatureValue* fields will contain the certificate signature. The **gsk\_encode\_signature()** routine can be used to recreate the encoded CRL from the *x509\_crl* structure returned by the **gsk\_decode\_crl()** routine.

Character strings will be returned using UTF-8 encoding. If necessary, the application can call **iconv()** to convert the string to a different encoding.

The certificate extensions will be returned with the extension values in ASN.1 encoded format. The **gsk\_decode\_certificate\_extension()** routine can be called to decode a particular certificate extension. This allows all of the certificate extensions to be returned even when one or more extensions cannot be processed by the System SSL runtime.

## `gsk_decode_name()`

---

### `gsk_decode_name()`

Decodes a X.509 name.

### Format

```
#include <gskcms.h>

gsk_status gsk_decode_name (
    gsk_buffer *    stream,
    x509_name *    name)
```

### Parameters

*stream*

Specifies the ASN.1 stream for the name.

*name*

Returns the decoded X.509 name. The application should release the name when it is no longer needed by calling the `gsk_free_name()` routine.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following is a common error:

#### **[ASN\_NO\_MEMORY]**

Insufficient memory is available.

### Usage

The `gsk_decode_name()` routine will decode an ASN.1 DER-encoded X.509 name. The name must have been encoded as described in RFC 2459 (Internet X.509 Public Key Infrastructure). Character strings will be stored in UTF-8 format and the *stringType* field in the `x509_rdn_attribute` structure will be set to indicate the ASN.1 encoded string type.

## gsk\_delete\_record()

Deletes a record from a key or request database.

### Format

```
#include <gskcms.h>

gsk_status gsk_delete_record (
    gsk_handle    db_handle,
    gsk_int32     record_id)
```

### Parameters

*db\_handle*

Specifies the database handle return by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine.

*record\_id*

Specifies the database record to be deleted.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BACKUP\_EXISTS]

The backup file already exists.

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_IO\_ERROR]

Unable to write record.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_RECORD\_NOT\_FOUND]

Record is not found.

#### [CMSERR\_SIGNED\_CERTS]

The database contains records signed using the certificate.

#### [CMSERR\_UPDATE\_NOT\_ALLOWED]

Database is not open for update.

### Usage

The **gsk\_delete\_record()** routine deletes a record from a key or request database. The database must be open for update in order to delete records. The unique record identifier identifies the record to be deleted. A certificate record cannot be deleted from a key database if the database contains records that were signed using the certificate.

The database file is updated as part of the **gsk\_delete\_record()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## `gsk_dn_to_name()`

---

### `gsk_dn_to_name()`

Converts a DN string to a X.509 name.

#### Format

```
#include <gskcms.h>

gsk_status gsk_dn_to_name (
    const char *    dn,
    x509_name *    name)
```

#### Parameters

*dn* Specifies the distinguished name in the local code page.

*name*

Returns the X.509 name. The X.509 strings use UTF-8 encoding. The application should call the `gsk_free_name()` routine to release the name when it is no longer needed.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

##### [ASN\_ATTR\_NOT\_FOUND]

An attribute type is not recognized.

##### [ASN\_CANT\_CONVERT]

An encoded attribute value contains characters from the wrong character set.

##### [ASN\_INVALID\_VALUE]

An attribute value is not valid.

##### [ASN\_NO\_MEMORY]

Insufficient storage is available.

##### [ASN\_WRONG\_TYPE]

An encoded attribute value does not represent a character string.

##### [ASN\_X500\_NO\_AVA\_SEP]

An attribute value separator is missing.

##### [ASN\_X500\_NO\_VALUE]

An attribute value is missing.

##### [ASN\_X500\_OID\_SYNTAX\_ERROR]

An object identifier is not valid.

##### [ASN\_X500\_SYNTAX\_ERROR]

The DN string format is not valid.

#### Usage

The `gsk_dn_to_name()` routine converts a distinguished name (DN) string to a X.509 name in accordance with RFC 2253 (UTF-8 String Representation of Distinguished Names). The input string consists of single-byte characters in the local code page. A double-byte character is represented using the escaped UTF-8 encoding of the double-byte character in the Unicode character set.

Attribute types may be specified using either attribute names or numeric object identifiers. Attribute values must represent string values.

## gsk\_dn\_to\_name()

The following DN attribute names are recognized by the System SSL runtime. An error will be returned if the DN contains an unrecognized attribute name.

<b>C</b>	Country
<b>CN</b>	Common name
<b>DC</b>	Domain component
<b>E</b>	E-mail address
<b>EMAIL</b>	E-mail address (preferred)
<b>EMAILADDRESS</b>	E-mail address
<b>L</b>	Locality
<b>O</b>	Organization name
<b>OU</b>	Organizational unit name
<b>PC</b>	Postal code
<b>S</b>	State or province
<b>SN</b>	Surname
<b>SP</b>	State or province
<b>ST</b>	State or province (preferred)
<b>STREET</b>	Street
<b>T</b>	Title

The following is an example of a DN using attribute names and string values:

```
CN=Ronald Hoffman,OU=Endicott,O=IBM,C=US
```

The following is the same DN using object identifiers and encoded string values. The encoded string values represent the ASN.1 DER encoding of the string. The System SSL runtime supports these ASN.1 string types: PRINTABLE, VISIBLE, TELETEX, IA5, UTF8, BMP, and UCS.

```
2.5.4.3=#130E526F6E616C6420486F666666D616E,2.5.4.11=#1308456E6469636F7474,  
2.5.4.10=#130349424D,2.5.4.6=13025553
```

Individual characters can be represented using escape sequences. This is useful when the character cannot be represented in a single-byte character set. The hexadecimal value for the escape sequence is the UTF-8 encoding of the character in the Unicode character set.

Unicode Letter Description	10646 code	UTF-8	Quoted
LATIN CAPITAL LETTER L	U0000004C	0x4C	L
LATIN SMALL LETTER U	U00000075	0x75	u
LATIN SMALL LETTER C WITH CARON	U0000010D	0xC48D	\C4\8D
LATIN SMALL LETTER I	U00000069	0x69	i
LATIN SMALL LETTER C WITH ACUTE	U00000107	0xC487	\C4\87

```
SN=Lu\C4\8Di\C4\87
```

An escape sequence can also be used for special characters which are part of the name and are not to be interpreted as delimiters. For example:

```
CN=L. Eagle,OU=Jones\, Dale and Mian,O=IBM,C=US
```

## **gsk\_encode\_base64()**

---

### **gsk\_encode\_base64()**

Encodes binary data using Base64 encoding.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_encode_base64 (
    gsk_buffer *    input_data,
    gsk_buffer *    encoded_data)
```

#### **Parameters**

*input\_data*

Specifies the data to be encoded.

*encoded\_data*

Returns the encoded stream in the local code page. The application should call the **gsk\_free\_buffer()** routine to release the encoded stream when it is no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following is a common error:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **Usage**

The **gsk\_encode\_base64()** routine will encode binary data using Base64 encoding. The encoded stream will consist of printable characters in the local code page. A new line will be inserted after each group of 64 encoded characters with a final new line at the end of the encoded stream. The **gsk\_decode\_base64()** routine can be used to decode the data.

## gsk\_encode\_certificate\_extension()

Encodes a X.509 certificate extension.

### Format

```
#include <gskcms.h>

gsk_status gsk_encode_certificate_extension (
    x509_decoded_extension *   decoded_extension,
    gsk_boolean                critical,
    x509_extension *          encoded_extension)
```

### Parameters

*decoded\_extension*

Specifies the decoded extension data.

*critical*

Specify TRUE if this is a critical extension or FALSE if it is not a critical extension.

*encoded\_extension*

Returns the encoded X.509 extension. The application should call the **gsk\_free\_certificate\_extension()** routine to release the extension when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [ASN\_NO\_MEMORY]

Insufficient memory is available.

#### [CMSERR\_EXT\_NOT\_SUPPORTED]

The certificate extension is not supported.

#### [CMSERR\_NO\_MEMORY]

Insufficient memory is available.

### Usage

The **gsk\_encode\_certificate\_extension()** routine encodes a certificate extension and returns the encoded extension in a format that can be used as input to the **gsk\_encode\_certificate()** routine.

The **gsk\_encode\_certificate\_extension()** routine assumes character strings use UTF-8 encoding. The application is responsible for providing character data in this format.

The following certificate extensions are supported:

- AuthorityInfoAccess
- AuthorityKeyIdentifier
- BasicConstraints
- CertificateIssuer
- CertificatePolicies
- CrlDistributionPoints
- CrlNumber
- CrlReasonCode
- DeltaCrlIndicator
- ExtKeyUsage

## **gsk\_encode\_certificate\_extension()**

- | • HoldInstructionCode
- | • InvalidityDate
- | • IssuerAltName
- | • IssuingDistributionPoint
- | • KeyUsage
- | • NameConstraints
- | • PolicyConstraints
- | • PolicyMappings
- | • PrivateKeyUsagePeriod
- | • SubjectAltName
- | • SubjectDirectoryAttributes
- | • SubjectKeyIdentifier

| The following general name types are supported:

- | • DirectoryName
- | • DnsName
- | • IpAddress
- | • RegisteredId
- | • Rfc822Name
- | • UniformResourceIdentifier

| Refer to RFC 2459 (Internet X.509 Public Key Infrastructure) for more information about the various certificate extensions.

## gsk\_encode\_name()

Encodes a X.509 name.

### Format

```
#include <gskcms.h>

gsk_status gsk_encode_name (
    x509_name *    name,
    gsk_buffer *  stream)
```

### Parameters

*name*

Specifies X.509 name.

*stream*

Returns the ASN.1 stream for the name. The application should release the stream when it is no longer needed by calling the **gsk\_free\_buffer()** routine.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [ASN\_CANT\_CONVERT]

A character string contains characters not allowed for the string type.

#### [ASN\_NO\_MEMORY]

Insufficient memory is available.

### Usage

The **gsk\_encode\_name()** routine will encode a X.509 name as an ASN.1 stream. The name will be encoded as described in RFC 2459 (Internet X.509 Public Key Infrastructure).

The *stringType* field in the *x509\_rdn\_attribute* structure will be used to determine the format for an encoded directory string. If it is set to *x509\_string\_unknown*, the **gsk\_encode\_name()** routine will attempt to encode the string as an ASN.1 printable string. If the string contains characters not included in the printable string set, the string will be encoded as an ASN.1 UTF-8 string. There are a couple of mandatory exceptions:

- The *countryName* attribute is always encoded as a printable string
- The *dnQualifier* attribute is always encoded as a printable string
- The *emailAddress* attribute is always encoded as an IA5 string
- The *domainComponent* attribute is always encoded as an IA5 string

## `gsk_encode_signature()`

---

### `gsk_encode_signature()`

Encodes an ASN.1 stream and the accompanying signature.

#### Format

```
#include <gskcms.h>

gsk_status gsk_encode_signature (
    gsk_buffer *                unsigned_stream,
    x509_algorithm_identifier * algorithm,
    gsk_bitstring *            signature,
    gsk_buffer *                signed_stream)
```

#### Parameters

*unsigned\_stream*

Specifies the unsigned ASN.1 stream.

*algorithm*

Specifies the algorithm used to compute the signature.

*signature*

Specifies the signature for the ASN.1 stream.

*signed\_stream*

Returns the encoded signature stream. The application should call the **`gsk_free_buffer()`** routine to release the encoded stream when it is no longer needed.

#### Results

The function return value will be 0 (**`GSK_OK`**) if no error is detected. Otherwise, it will be one of the return codes listed in the **`gskcms.h`** include file. The following is a common error:

**[ASN\_NO\_MEMORY]**

Insufficient memory is available.

#### Usage

The **`gsk_encode_signature()`** routine is used to encode an unsigned ASN.1 stream and the digital signature generated for the stream. The signature is encoded using ASN.1 DER (Distinguished Encoding Rules). The application is responsible for ensuring the validity of the supplied information.

## gsk\_export\_certificate()

Exports a certificate.

### Format

```
#include <gskcms.h>

gsk_status gsk_export_certificate (
    gsk_handle          db_handle,
    const char *        label,
    gskdb_export_format format,
    gsk_buffer *        stream)
```

### Parameters

#### *db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine. The database must be a key database and not a request database.

#### *label*

Specifies the label for the database record. The label is specified in the local code page.

#### *format*

Specifies the export format. The following values may be specified:

#### **gskdb\_export\_der\_binary**

Binary ASN.1 DER-encoded

#### **gskdb\_export\_der\_base64**

Base64 ASN.1 DER-encoded

#### **gskdb\_export\_pkcs7\_binary**

Binary PKCS #7 Cryptographic Message Syntax

#### **gskdb\_export\_pkcs7\_base64**

Base64 PKCS #7 Cryptographic Message Syntax

#### *stream*

Return the byte stream for the encoded certificate. The application should call the **gsk\_free\_buffer()** routine to release the storage when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

#### **[CMSERR\_BAD\_LABEL]**

No database record label is supplied.

#### **[CMSERR\_FMT\_NOT\_SUPPORTED]**

An unsupported export file format is specified.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **[CMSERR\_RECORD\_NOT\_FOUND]**

The requested record is not found.

## **gsk\_export\_certificate()**

### **Usage**

The **gsk\_export\_certificate()** routine exports a X.509 certificate. The certificate can be exported using either the ASN.1 DER encoding for the certificate or the Cryptographic Message Syntax (PKCS #7) encoding for the certificate. This can be either the binary value or the Base64 encoding of the binary value. A Base64 encoded stream will be in the local code page and will include the encoding header and footer lines.

## gsk\_export\_certification\_request()

Exports a PKCS #10 certification request.

### Format

```
#include <gskcms.h>

gsk_status gsk_export_certification_request (
    gsk_handle          db_handle,
    const char *        label,
    gskdb_export_format format,
    gsk_buffer *        stream)
```

### Parameters

#### *db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine. The database must be a request database and not a key database.

#### *label*

Specifies the label for the database record. The label is specified in the local code page.

#### *format*

Specifies the export format. The following values may be specified:

#### **gskdb\_export\_der\_binary**

Binary ASN.1 DER-encoded

#### **gskdb\_export\_der\_base64**

Base64 ASN.1 DER-encoded

#### *stream*

Return the byte stream for the encoded certification request. The application should call the **gsk\_free\_buffer()** routine to release the storage when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

#### **[CMSERR\_BAD\_LABEL]**

No database record label is supplied.

#### **[CMSERR\_FMT\_NOT\_SUPPORTED]**

An unsupported export file format is specified.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certification requests.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **[CMSERR\_RECORD\_NOT\_FOUND]**

The requested record is not found.

### Usage

The **gsk\_export\_certification\_request()** routine exports a PKCS #10 certification request. The request can be exported using either the ASN.1 DER encoding for the request or the Base64 encoding of the binary value. A Base64 encoded stream will be in the local code page and will include the encoding

## **gsk\_export\_certification\_request()**

| header and footer lines.

## gsk\_export\_key()

Exports a certificate and the associated private key.

### Format

```
#include <gskcms.h>

gsk_status gsk_export_key (
    gsk_handle          db_handle,
    const char *        label,
    gskdb_export_format format,
    x509_algorithm_type algorithm,
    const char *        password,
    gsk_buffer *        stream)
```

### Parameters

#### *db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine. The database must be a key database and not a request database. For a SAF key ring database, the private key must be stored in the SAF database and not in ICSF.

#### *label*

Specifies the label for the database record. The label is specified in the local code page.

#### *format*

Specifies the export format. The following values may be specified:

**gskdb\_export\_pkcs12v1\_binary**  
Binary PKCS #12 Version 1

**gskdb\_export\_pkcs12v1\_base64**  
Base64 PKCS #12 Version 1

**gskdb\_export\_pkcs12v3\_binary**  
Binary PKCS #12 Version 3

**gskdb\_export\_pkcs12v3\_base64**  
Base64 PKCS #12 Version 3

#### *algorithm*

Specifies the encryption algorithm for the export file. The strong encryption algorithms may not be available depending upon government export regulations.

The following values may be specified for the PKCS #12 Version 1 format:

**x509\_alg\_pb1WithSha1And40BitRc2Cbc**  
40-bit RC2 with SHA-1 digest

**x509\_alg\_pb1WithSha1And128BitRc2Cbc**  
128bit RC2 with SHA-1 digest

**x509\_alg\_pb1WithSha1And40BitRc4**  
40-bit RC4 with SHA-1 digest

**X509\_alg\_pb1WithSha1And128BitRc4**  
128-bit RC4 with SHA-1 digest

**x509\_alg\_pb1WithSha1And3DesCbc**  
Triple DES with SHA-1 digest

The following values may be specified for the PKCS #12 Version 3 format:

## **gsk\_export\_key()**

**x509\_alg\_pbeWithSha1And40BitRc2Cbc**

40-bit RC2 with SHA-1 digest

**x509\_alg\_pbeWithSha1And128BitRc2Cbc**

128bit RC2 with SHA-1 digest

**x509\_alg\_pbeWithSha1And40BitRc4**

40-bit RC4 with SHA-1 digest

**X509\_alg\_pbeWithSha1And128BitRc4**

128-bit RC4 with SHA-1 digest

**x509\_alg\_pbeWithSha1And3DesCbc**

Triple DES with SHA-1 digest

### *password*

Specifies the password for the export file. The password is in the local code page and must consist of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string. The user will be prompted to enter the password if NULL is specified for this parameter.

### *stream*

Return the byte stream for the encoded certificate. The application should call the **gsk\_free\_buffer()** routine to release the storage when it is no longer needed.

## **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

### **[CMSERR\_BAD\_LABEL]**

No database record label is supplied.

### **[CMSERR\_FMT\_NOT\_SUPPORTED]**

An unsupported export file format is specified.

### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_NO\_PRIVATE\_KEY]**

The database entry does not contain a private key.

### **[CMSERR\_RECORD\_NOT\_FOUND]**

The requested record is not found.

## **Usage**

The **gsk\_export\_key()** routine exports a X.509 certificate and the associated private key. The certificate can be exported using either the PKCS #12 Version 1 format or the PKCS #12 Version 3 format. This can be either the binary value or the Base64 encoding of the binary value. A Base64 encoded stream will be in the local code page and will include the encoding header and footer lines.

The PKCS #12 Version 1 format is obsolete. However, it is the only format supported by some SSL implementations and must be used when moving a certificate and key to one of those systems. You should use either **x509\_alg\_pb1WithSha1And40BitRc2Cbc** or **x509\_alg\_pb1WithSha1And3DesCbc** for interoperability with these older SSL implementations.

---

## gsk\_free\_buffer()

Releases storage allocated for a buffer.

### Format

```
#include <gskcms.h>

void gsk_free_buffer (
    gsk_buffer *    buffer)
```

### Parameters

*buffer*

Specifies the buffer to be released. The gsk\_buffer structure will be initialized to zero upon completion.

### Usage

The **gsk\_free\_buffer()** routine is used to release storage allocated for a buffer.

## **gsk\_free\_certificate()**

---

### **gsk\_free\_certificate()**

Releases storage allocated for a X.509 certificate.

### **Format**

```
#include <gskcms.h>

void gsk_free_certificate (
    x509_certificate *    certificate)
```

### **Parameters**

*certificate*

Specifies the certificate to be released. The x509\_certificate structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_certificate()** routine is used to release storage allocated for an X.509 certificate.

---

## gsk\_free\_certificates()

Releases storage allocated for an array of certificates.

### Format

```
#include <gskcms.h>
```

```
void gsk_free_certificates (  
    pkcs_certificates *    certificates)
```

### Parameters

*certificate*

Specifies the certificate to be released. The `pkcs_certificates` structure will be initialized to zero upon completion.

### Usage

The `gsk_free_certificates()` routine is used to release storage allocated for an array of certificates.

## **gsk\_free\_certificate\_extension()**

---

### **gsk\_free\_certificate\_extension()**

Releases storage allocated for a X.509 certificate extension.

### **Format**

```
#include <gskcms.h>
```

```
void gsk_free_certificate_extension (  
                                     x509_extension * extension)
```

### **Parameters**

*extension*

Specifies the certificate extension to be released. The x509\_extension structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_certificate\_extension()** routine is used to release storage allocated for a X.509 certificate extension.

---

## gsk\_free\_certification\_request()

Releases storage allocated for a PKCS certification request.

### Format

```
#include <gskcms.h>

void gsk_free_certification_request (
    pkcs_cert_request * request)
```

### Parameters

*request*

Specifies the certification request to be released. The `pkcs_cert_request` structure will be initialized to zero upon completion.

### Usage

The `gsk_free_certification_request()` routine is used to release storage allocated for a Public Key Cryptography Standards (PKCS) certification request.

## **gsk\_free\_content\_info()**

---

### **gsk\_free\_content\_info()**

Releases storage allocated for PKCS #7 content information.

### **Format**

```
#include <gskcms.h>

void gsk_free_content_info (
    pkcs_content_info *    content_info)
```

### **Parameters**

*content\_info*

Specifies the content information to be released. The `pkcs_content_info` structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_content\_info()** routine is used to release storage allocated for a Public Key Cryptography Standards (PKCS) content information.

---

## gsk\_free\_crl()

Releases storage allocated for a X.509 certificate revocation list.

### Format

```
#include <gskcms.h>

void gsk_free_crl (
    x509_crl *    crl)
```

### Parameters

*crl* Specifies the certificate revocation list to be released. The x509\_crl structure will be initialized to zero upon completion.

### Usage

The **gsk\_free\_crl()** routine is used to release storage allocated for a X.509 certificate revocation list.

## **gsk\_free\_crls()**

---

### **gsk\_free\_crls()**

Releases storage allocated for an array of X.509 certificate revocation lists.

### **Format**

```
#include <gskcms.h>

void gsk_free_crls (
    x509_crls *    crls)
```

### **Parameters**

*crls*

Specifies the array of certificate revocation lists to be released. The x509\_crls structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_crls()** routine is used to release storage allocated for an array of X.509 certificate revocation lists.

## **gsk\_free\_decoded\_extension()**

Frees a decoded certificate extension.

### **Format**

```
#include <gskcms.h>
```

```
void gsk_free_decoded_extension (  
    x509_decoded_extension *    decoded_extension)
```

### **Parameters**

*decoded\_extension*

Specifies the certificate extension to be released. The `x509_decoded_extension` structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_decoded\_extension()** routine is used to release storage allocated for a decoded X.509 certificate extension.

## **gsk\_free\_name()**

---

### **gsk\_free\_name()**

Releases storage allocated for a X.509 name.

### **Format**

```
#include <gskcms.h>

void gsk_free_name (
    x509_name *    name)
```

### **Parameters**

*name*

Specifies the name to be released. The x509\_name structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_name()** routine is used to release storage allocated for a X.509 name.

---

## gsk\_free\_private\_key\_info()

Releases storage allocated for private key information.

### Format

```
#include <gskcms.h>
```

```
void gsk_free_private_key_info (  
                                pkcs_private_key_info *   info)
```

### Parameters

*info*

Specifies the private key information to be released. The `pkcs_private_key_info` structure will be initialized to zero upon completion.

### Usage

The `gsk_free_private_key_info()` routine is used to release storage allocated for private key information.

## **gsk\_free\_public\_key\_info()**

---

### **gsk\_free\_public\_key\_info()**

Releases storage allocated for public key information.

### **Format**

```
#include <gskcms.h>
```

```
void gsk_free_public_key_info (
                                x509_public_key_info *   info)
```

### **Parameters**

*info*

Specifies the public key information to be released. The x509\_public\_key\_info structure will be initialized to zero upon completion.

### **Usage**

The **gsk\_free\_public\_key\_info()** routine is used to release storage allocated for public key information.

---

## gsk\_free\_record()

Releases storage allocated for a database record.

### Format

```
#include <gskcms.h>

void gsk_free_record (
    gskdb_record *    record)
```

### Parameters

*record*

Specifies the database record to be released. The gskdb\_record structure is released in addition to the record data.

### Usage

The **gsk\_free\_record()** routine is used to release storage allocated for a database record.

## **gsk\_free\_records()**

---

### **gsk\_free\_records()**

Releases storage allocated for an array of database records.

### **Format**

```
#include <gskcms.h>

void gsk_free_records (
    int          num_records,
    gskdb_record ** records)
```

### **Parameters**

*num\_records*

Specifies the number of records in the array.

*records*

Specifies the database record array to be released. The gskdb\_record structures are released in addition to the record data.

### **Usage**

The **gsk\_free\_records()** routine is used to release storage allocated for an array of database records.

---

## gsk\_free\_string()

Releases storage allocated for a string.

### Format

```
#include <gskcms.h>

void gsk_free_string (
    char *    string)
```

### Parameters

*string*  
Specifies the string to be released.

### Usage

The `gsk_free_string()` routine is used to release storage allocated for a string.

## **gsk\_free\_strings()**

---

### **gsk\_free\_strings()**

Releases storage allocated for an array of strings.

### **Format**

```
#include <gskcms.h>

void gsk_free_strings (
    int          num_strings,
    char **     strings)
```

### **Parameters**

*num\_strings*

Specifies the number of strings in the array.

*strings*

Specifies the array of strings to be released.

### **Usage**

The **gsk\_free\_strings()** routine is used to release storage allocated for an array of strings.

---

## gsk\_generate\_random\_bytes()

Generates a random byte stream.

### Format

```
#include <gskcms.h>

void gsk_generate_random_bytes (
    gsk_buffer *    buffer)
```

### Parameters

*buffer*

Specifies the buffer for the random byte stream. The application is responsible for providing the buffer and setting the *length* and *data* fields appropriately.

### Usage

The **gsk\_generate\_random\_bytes()** routine will generate a random byte stream. The application provides the buffer for the byte stream. The length value determines how many bytes will be generated. The generated byte stream will not contain any zero bytes.

## **gsk\_get\_cms\_vector()**

---

### **gsk\_get\_cms\_vector()**

Obtains the address of the Certificate Management Services function vector.

#### **Format**

```
#include <gskcms.h>

void gsk_get_cms_vector (
    gsk_uint32 *          function_mask,
    gsk_cms_vector **    function_vector)
```

#### **Parameters**

*function\_mask*

Returns a bit mask indicating the Certificate Management Services level.

*function\_vector*

Returns the address of the Certificate Management Services function vector.

#### **Usage**

Certificate Management Services (CMS) functions can be called using either static binding or runtime binding. Static binding is performed when the application is compiled while runtime binding is performed when the application is run.

In order to use static binding, the CMS sidefile is specified as input to the binder. This causes all CMS functions to be resolved at bind time and will cause the CMS DLL to be implicitly loaded when the application is run.

In order to use runtime binding, the CMS DLL must be explicitly loaded by the application and the CMS functions must be called using indirect addresses. The **gsk\_get\_cms\_vector()** routine allows an application to obtain the address of the CMS function vector containing an entry for each CMS API routine. This eliminates the need for the application to build the function vector through repeated calls to the **dllqueryfn()** routine.

The function mask indicates the capabilities of the version of the CMS DLL. The following values have been defined:

#### **GSKCMS\_API\_LVL1**

CMS functions provided as part of z/OS Version 1 Release 4 are available.

## gsk\_get\_default\_key()

Gets the default key record.

### Format

```
#include <gskcms.h>

gsk_status gsk_get_default_key (
    gsk_handle      db_handle,
    gskdb_record ** record)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*record*

Returns the database record. The application should call the **gsk\_free\_record()** routine to release the record when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_RECORD\_DELETED]

The requested record is deleted.

#### [CMSERR\_RECORD\_NOT\_FOUND]

There is no default key for the database.

### Usage

The **gsk\_get\_default\_key()** routine retrieves the record for the default key. An error will be returned if there is no default key.

## **gsk\_get\_default\_label()**

---

### **gsk\_get\_default\_label()**

Gets the label of the default key record.

### **Format**

```
#include <gskcms.h>

gsk_status gsk_get_default_label (
    gsk_handle db_handle,
    char **    label)
```

### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*label*

Returns the label of the default key record. The application should call the **gsk\_free\_string()** routine to release the label when it is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **[CMSERR\_RECORD\_DELETED]**

The requested record is deleted.

#### **[CMSERR\_RECORD\_NOT\_FOUND]**

There is no default key for the database.

### **Usage**

The **gsk\_get\_default\_label()** routine returns the label of the default key record. An error will be returned if there is no default key.

## gsk\_get\_directory\_certificates()

Gets the certificates stored in the LDAP directory for the subject.

### Format

```
#include <gskcms.h>

gsk_status gsk_get_directory_certificates (
    gsk_handle          directory_handle,
    x509_name *        subject_name,
    gsk_boolean        ca_certificates,
    pkcs_certificates * certificates)
```

### Parameters

*directory\_handle*

Specifies the directory handle returned by the **gsk\_open\_directory()** routine.

*subject\_name*

Specifies the certificate subject.

*ca\_certificates*

Specify TRUE if the subject is a certification authority or FALSE if the subject is an end entity.

*certificates*

Returns the certificates for the subject. The application should call the **gsk\_free\_certificates()** routine to release the certificates when they are no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BAD\_HANDLE]

The directory handle is not valid.

#### [CMSERR\_LDAP]

An error is detected by the LDAP runtime support.

#### [CMSERR\_LDAP\_NOT\_AVAILABLE]

The LDAP server is not available.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_RECORD\_NOT\_FOUND]

The requested certificate is not found.

### Usage

The **gsk\_get\_directory\_certificates()** routine retrieves the certificates stored in the LDAP directory for the specified subject name. The directory schema is defined by RFC 2587 (PKIX LDAPV2 Schema). The certificates are stored as attributes of the subject directory entry. Each certificate is encoded as defined by RFC 2459 (X.509 Public Key Infrastructure). The *userCertificate* attribute is used to retrieve end entity certificates while the *caCertificate* attribute is used to retrieve certification authority certificates.

Retrieved certificates are cached so that it is not necessary to contact the LDAP server for subsequent requests for the same certificates. The cached certificates will be released when the **gsk\_close\_directory()** routine is called to close the directory handle.

## `gsk_get_directory_crls()`

---

### `gsk_get_directory_crls()`

Gets the certificate revocation lists stored in the LDAP directory for the issuer.

#### Format

```
#include <gskcms.h>

gsk_status gsk_get_directory_crls (
    gsk_handle      directory_handle,
    x509_name *     dist_point_name,
    x509_name *     issuer_name,
    gsk_boolean     ca_lists,
    x509_crls *     crls)
```

#### Parameters

*directory\_handle*

Specifies the directory handle returned by the `gsk_open_directory()` routine.

*dist\_point\_name*

Specifies the CRL distribution point name.

*issuer\_name*

Specifies the CRL issuer name.

*ca\_lists*

Specify TRUE to retrieve the revocation lists for CA certificates or FALSE to retrieve the revocations lists for end entity certificates.

*crls*

Returns the certificate revocation lists. The application should call the `gsk_free_crls()` routine to release the lists when they are no longer needed.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

##### [CMSERR\_BAD\_HANDLE]

The directory handle is not valid.

##### [CMSERR\_LDAP]

An error is detected by the LDAP runtime support.

##### [CMSERR\_LDAP\_NOT\_AVAILABLE]

The LDAP server is not available.

##### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

##### [CMSERR\_RECORD\_NOT\_FOUND]

The requested CRL is not found.

#### Usage

The `gsk_get_directory_crls()` routine retrieves the certificate revocation lists (CRLs) stored in the LDAP directory for the specified issuer name. The directory schema is defined by RFC 2587 (PKIX LDAPV2 Schema). The revocation lists are stored as attributes of the issuer directory entry. Each CRL is encoded as defined by RFC 2459 (X.509 Public Key Infrastructure). The *certificateRevocationList* attribute is used to retrieve revocation lists for end entity certificates while the *authorityRevocationList* attribute is used to retrieve revocation lists for certification authority certificates.

## **gsk\_get\_directory\_crls()**

| The *dist\_point\_name* parameter specifies the CRL distribution point name. This name is used as the distinguished name for the LDAP directory entry. The *issuer\_name* parameter specifies the CRL issuer name. This name must match the issuer name stored in the CRL.

| Retrieved certificate revocation lists are cached so that it is not necessary to contact the LDAP server for subsequent requests for the same issuer. The cached revocation lists will be released when the **gsk\_close\_directory()** routine is called to close the directory handle. The cached entries will also be discarded at the end of the cache timeout specified by the GSK\_CRL\_CACHE\_TIMEOUT environment variable (the default timeout is 24 hours).

## **gsk\_get\_record\_by\_id()**

---

### **gsk\_get\_record\_by\_id()**

Gets a database record using the record identifier.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_get_record_by_id (
    gsk_handle          db_handle,
    gsk_int32           record_id,
    gskdb_record **    record)
```

#### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*record\_id*

Specifies the record identifier.

*record*

Returns the database record. The application should call the **gsk\_free\_record()** routine to release the record when it is no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

##### **[CMSERR\_RECORD\_NOT\_FOUND]**

The requested record is not found.

#### **Usage**

The **gsk\_get\_record\_by\_id()** routine retrieves a record from a key or request database based upon the unique record identifier. The record identifier is assigned when the record is added to the database and does not change as records are added and deleted.

## gsk\_get\_record\_by\_index()

Gets a database record using a sequential index.

### Format

```
#include <gskcms.h>

gsk_status gsk_get_record_by_index (
    gsk_handle          db_handle,
    int                 index,
    gskdb_record **    record)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*index*

Specifies the sequential index of the record. The first record in the database is record 1.

*record*

Returns the database record. The application should call the **gsk\_free\_record()** routine to release the record when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_RECORD\_NOT\_FOUND]

The requested record is not found.

### Usage

The **gsk\_get\_record\_by\_index()** routine retrieves a record from a key or request database based upon a sequential index number. The first record in the database is record 1. The index numbers will change as records are added and deleted.

## **gsk\_get\_record\_by\_label()**

---

### **gsk\_get\_record\_by\_label()**

Gets a database record using the record label.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_get_record_by_label (
    gsk_handle          db_handle,
    const char *        label,
    gskdb_record **    record)
```

#### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*label*

Specifies the label of the database record. The label is specified in the local code page.

*record*

Returns the database record. The application should call the **gsk\_free\_record()** routine to release the record when it is no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

##### **[CMSERR\_RECORD\_NOT\_FOUND]**

The requested record is not found.

#### **Usage**

The **gsk\_get\_record\_by\_label()** routine retrieves a record from a key or request database based upon the record label. The record label is a character string assigned when the record is added to the database. The label comparison is case sensitive.

## gsk\_get\_record\_by\_subject()

Gets one or more database records using the certificate subject.

### Format

```
#include <gskcms.h>

gsk_status gsk_get_record_by_subject (
    gsk_handle          db_handle,
    x509_name *        name,
    int *               num_records,
    gskdb_record ***   records)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*name*

Specifies the certificate subject.

*num\_records*

Returns the number of records in the array.

*records*

Returns the array of database records. The application should call the **gsk\_free\_records()** routine to release the array when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_INCORRECT\_DBTYPE]

The database does not support this operation.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_RECORD\_NOT\_FOUND]

The requested record is not found.

### Usage

The **gsk\_get\_record\_by\_subject()** routine retrieves all records from a key database with the specified subject name.

## **gsk\_get\_record\_labels()**

---

### **gsk\_get\_record\_labels()**

Gets the record labels for a key or request database.

#### **Format**

```
#include <gskcms.h>

gsk_status gsk_get_record_labels (
    gsk_handle     db_handle,
    gsk_boolean    private_key,
    int *          num_labels,
    char ***       labels)
```

#### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*private\_key*

Specify TRUE if labels for records containing a private key are to be returned. Specify FALSE if labels for records without a private key are to be returned.

*num\_labels*

Returns the number of record labels.

*labels*

Returns an array of string addresses. The labels are returned using the local code page. The application should call the **gsk\_free\_strings()** routine to release the record labels when they are no longer needed.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **Usage**

The **gsk\_get\_record\_labels()** routine returns all of the record labels for a key or request database. The **gsk\_get\_record\_by\_label()** routine can then be used to retrieve a specific database record. The array address will be set to NULL and the number of labels will be set to 0 if there are no records in the database.

## gsk\_get\_update\_code()

Gets the database update code.

### Format

```
#include <gskcms.h>

gsk_status gsk_get_update_code (
    gsk_handle      db_handle,
    gsk_uint32 *    update_code)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine.

*update\_code*

Returns the current update code for the database.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_get\_update\_code()** routine returns the current update code for the database. For a file-based database, this is the modification timestamp. For a SAF key ring, this is the ring sequence number. If an update has occurred, the application can close and then re-open the database to pick up the updates.

## **gsk\_import\_certificate()**

---

### **gsk\_import\_certificate()**

Imports a certificate.

#### **Format**

```
#include <gskcms.h>
gsk_status gsk_import_certificate (
    gsk_handle      db_handle,
    const char *    label,
    gsk_buffer *    stream)
```

#### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine.

*label*

Specifies the label for the new database record. The label is specified in the local code page.

*stream*

Specifies the byte stream of the encoded certificate.

#### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

##### **[CMSERR\_BAD\_BASE64\_ENCODING]**

The Base64 encoding of the import file is not correct.

##### **[CMSERR\_BAD\_ENCODING]**

The import file format is not recognized.

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

##### **[CMSERR\_BAD\_SIGNATURE]**

The certificate signature is not correct.

##### **[CMSERR\_DUPLICATE\_CERTIFICATE]**

The database already contains the certificate.

##### **[CMSERR\_EXPIRED]**

The certificate is expired.

##### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

##### **[CMSERR\_INCORRECT\_KEY\_USAGE]**

The issuer certificate does not allow signing certificates.

##### **[CMSERR\_ISSUER\_NOT\_CA]**

The certificate issuer is not a certification authority.

##### **[CMSERR\_ISSUER\_NOT\_FOUND]**

The issuer certificate is not in the key database.

**[CMSERR\_IO\_ERROR]**

Unable to write record.

**[CMSERR\_LABEL\_NOT\_UNIQUE]**

The record label is not unique.

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

**[CMSERR\_NOT\_YET\_VALID]**

The certificate is not yet valid.

**[CMSERR\_RECORD\_TOO\_BIG]**

The record is larger than the database record length.

**[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

## Usage

The **gsk\_import\_certificate()** routine imports an X.509 certificate and creates a new database record. An error will be returned if the certificate is already in the database. The database must be a key database and must be open for update in order to import certificates.

The supplied stream can represent either the ASN.1 DER encoding for the certificate or the Cryptographic Message Syntax (PKCS #7) encoding for the certificate. This can be either the binary value or the Base64 encoding of the binary value. A Base64 encoded stream must be in the local code page and must include the encoding header and footer lines.

The **gsk\_import\_certificate()** routine imports a single certificate. If the PKCS #7 message contains multiple certificates, only the first certificate will be imported.

A unique record identifier is assigned when the record is added to the database. The certificate signature will be verified using the certificate of the issuer. An error will be returned if the issuer certificate is not already in the key database. The certificate will be marked as a trusted certificate when it is added to the database.

The record label is used as a friendly name for the database entry. It can be any value and consists of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string.

An existing certificate can be replaced by specifying the label of the existing certificate. The issuer name, subject name, and subject public key in the new certificate must be the same as the existing certificate. If the existing certificate has a private key, the private key is not changed when the certificate is replaced.

The database file is updated as part of the **gsk\_import\_certificate()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## `gsk_import_key()`

---

### `gsk_import_key()`

Imports a certificate and associated private key.

#### Format

```
#include <gskcms.h>

gsk_status gsk_import_key (
    gsk_handle      db_handle,
    const char *    label,
    const char *    password,
    gsk_buffer *    stream)
```

#### Parameters

##### *db\_handle*

Specifies the database handle returned by the `gsk_create_database()` routine or the `gsk_open_database()` routine.

##### *label*

Specifies the label for the new database record. The label is specified in the local code page.

##### *password*

Specifies the password for the import file. The password is in the local code page and must consist of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string. The user will be prompted to enter the password if NULL is specified for this parameter.

##### *stream*

Specifies the byte stream for the encoded certificate and private key.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

##### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

##### **[CMSERR\_BAD\_BASE64\_ENCODING]**

The Base64 encoding of the import file is not correct.

##### **[CMSERR\_BAD\_ENCODING]**

The import file format is not recognized.

##### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

##### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

##### **[CMSERR\_BAD\_SIGNATURE]**

The certificate signature is not correct.

##### **[CMSERR\_DUPLICATE\_CERTIFICATE]**

The database already contains the certificate.

##### **[CMSERR\_EXPIRED]**

The certificate is expired.

##### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

**[CMSERR\_INCORRECT\_KEY\_USAGE]**

The issuer certificate does not allow signing certificates.

**[CMSERR\_ISSUER\_NOT\_CA]**

The certificate issuer is not a certification authority.

**[CMSERR\_ISSUER\_NOT\_FOUND]**

The issuer certificate is not in the key database.

**[CMSERR\_IO\_ERROR]**

Unable to write record.

**[CMSERR\_LABEL\_NOT\_UNIQUE]**

The record label is not unique.

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

**[CMSERR\_NOT\_YET\_VALID]**

The certificate is not yet valid.

**[CMSERR\_RECORD\_TOO\_BIG]**

The record is larger than the database record length.

**[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

## Usage

The **gsk\_import\_key()** routine imports an X.509 certificate and its private key and creates a new database record. An error will be returned if the database already contains the certificate. The database must be open for update in order to import certificates.

The certificate and key must have been encoded according to the Personal Information Exchange Syntax (PKCS #12). The supplied stream can be the binary ASN.1 sequence or the Base64 encoding of the ASN.1 sequence. A Base64 encoded stream is assumed to be in the local code page and must include the encoding header and footer lines.

The record label is used as a friendly name for the database entry. It can be any value and consists of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be an empty string.

A unique record identifier is assigned when the record is added to the database. The certificate signature will be verified using the certificate of the issuer. The certificate will be marked as a trusted certificate when it is added to the database.

Each certificate in the certification chain will be imported if it is present in the import file. The certificate subject name will be used as the label for certificates added from the certification chain. A chain certificate will not be added to the database if the label is not unique or if the certificate is already in the database.

The database file is updated as part of the **gsk\_import\_key()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## `gsk_make_content_msg()`

---

### `gsk_make_content_msg()`

Creates a PKCS #7 content information message.

#### Format

```
#include <gskcms.h>

gsk_status gsk_make_content_msg (
    pkcs_content_info *   content_info,
    gsk_buffer *         stream)
```

#### Parameters

*content\_info*

Specifies the content information for the message.

*stream*

Returns the ASN.1 DER-encoded stream. The application should call the **`gsk_free_buffer()`** routine to release the stream when it is no longer needed.

#### Results

The function return value will be 0 (**`GSK_OK`**) if no error is detected. Otherwise, it will be one of the return codes listed in the **`gskcms.h`** include file. The following are some common errors:

##### **`[CMSERR_CONTENT_NOT_SUPPORTED]`**

The content type is not supported

##### **`[CMSERR_NO_MEMORY]`**

Insufficient storage is available

#### Usage

The **`gsk_make_content_msg()`** routine creates a PKCS #7 (Cryptographic Message Syntax) message using the supplied content information and returns the ASN.1 DER-encoded ContentInfo sequence. The message content type can be any of the types defined by the PKCS #7 specification. The **`gsk_read_content_msg()`** routine can be used to extract the content information from the stream.

---

## gsk\_make\_data\_content()

Creates PKCS #7 Data content information from application data.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_data_content (
    gsk_buffer *      data,
    pkcs_content_info * content_info)
```

### Parameters

*data*

Specifies the application data.

*content\_info*

Returns the Data content information. The application should call the **gsk\_free\_content\_info()** routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

**[CMSERR\_NO\_CONTENT\_DATA]**

The application data length is zero

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available

### Usage

The **gsk\_make\_data\_content()** routine creates PKCS #7 (Cryptographic Message Syntax) Data content information. The **gsk\_read\_data\_content()** routine can be used to extract the application data from the content information.

## **gsk\_make\_data\_msg()**

---

### **gsk\_make\_data\_msg()**

Creates a PKCS #7 Data message from application data.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_make_data_msg (
    gsk_buffer *      data,
    gsk_buffer *      stream)
```

### **Parameters**

*data*

Specifies the application data.

*stream*

Returns the ASN.1 DER-encoded stream. The application should call the **gsk\_free\_buffer()** routine to release the stream when it is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_NO\_CONTENT\_DATA]**

The application data length is zero

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available

### **Usage**

The **gsk\_make\_data\_msg()** routine creates a PKCS #7 (Cryptographic Message Syntax) Data message and returns the ASN.1 DER-encoded ContentInfo sequence. The message content type will be Data. The **gsk\_read\_data\_msg()** routine can be used to extract the application data from the stream.

Calling the **gsk\_make\_data\_msg()** routine is equivalent to calling the **gsk\_make\_data\_content()** routine followed by the **gsk\_make\_content\_msg()** routine.

## gsk\_make\_encrypted\_data\_content()

Creates PKCS #7 EncryptedData content information.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_encrypted_data_content (
    int                version,
    x509_algorithm_type pbe_algorithm,
    const char *       password,
    int                iterations,
    pkcs_content_info * content_data,
    pkcs_content_info * content_info)
```

### Parameters

*version*

Specifies the PKCS #7 EncryptedData version number. This must be 0.

*pbe\_algorithm*

Specifies the password-based encryption algorithm.

*password*

Specifies the encryption password as a null-terminated string in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

*iterations*

Specifies the number of iterations used to derive the encryption key from the password. It is recommended that iterations be specified as 1024 or greater.

*content\_data*

Specifies the EncryptedData content. This must be one of the content information types defined in PKCS #7.

*content\_info*

Returns the EncryptedData content information. The application should call the **gsk\_free\_content\_info()** routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

**[CMSERR\_ALG\_NOT\_AVAILABLE]**

Encryption algorithm is not available

**[CMSERR\_ALG\_NOT\_SUPPORTED]**

Encryption algorithm is not supported

**[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported

**[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available

**[CMSERR\_VERSION\_NOT\_SUPPORTED]**

The version is not valid

## **gsk\_make\_encrypted\_data\_content()**

### **Usage**

The **gsk\_make\_encrypted\_data\_content()** routine creates PKCS #7 (Cryptographic Message Syntax) EncryptedData content information. The data content type must be one of the types defined by PKCS #7. The **gsk\_read\_encrypted\_data\_content()** routine can be used to extract the content data from the content information.

The encryption key is derived from the password as described in PKCS #5 (Password-based Encryption) and PKCS #12 (Personal Information Exchange). The selected algorithm determines how the key is derived from the password.

The following password-based encryption algorithms are supported. The strong encryption algorithms may not be available depending upon government export regulations.

- x509\_alg\_pbeWithMd2AndDesCbc - 56-bit DES encryption with MD2 digest - {1.2.840.113549.1.5.1}
- x509\_alg\_pbeWithMd5AndDesCbc - 56-bit DES encryption with MD5 digest - {1.2.840.113549.1.5.3}
- x509\_alg\_pbeWithSha1AndDesCbc - 56-bit DES encryption with SHA-1 digest - {1.2.840.113549.1.5.10}
- x509\_alg\_pbeWithMd2AndRc2Cbc - 64-bit RC2 encryption with MD2 digest - {1.2.840.113549.1.5.4}
- x509\_alg\_pbeWithMd5AndRc2Cbc - 64-bit RC2 encryption with MD5 digest - {1.2.840.113549.1.5.6}
- x509\_alg\_pbeWithSha1AndRc2Cbc - 64-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.5.11}
- x509\_alg\_pbeWithSha1And40BitRc2Cbc - 40-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.6}
- x509\_alg\_pbeWithSha1And128BitRc2Cbc - 128-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.5}
- x509\_alg\_pbeWithSha1And40BitRc4 - 40-bit RC4 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.2}
- x509\_alg\_pbeWithSha1And128BitRc4 - 128-bit RC4 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.1}
- x509\_alg\_pbeWithSha1And3DesCbc - 168-bit 3DES encryption with SHA-1 digest - {1.2.840.113549.1.12.1.3}

## gsk\_make\_encrypted\_data\_msg()

Creates a PKCS #7 EncryptedData message from application data.

### Format

```
#include <gskcms.h>

gsk_status gsk_make_encrypted_data_msg (
    int                version,
    x509_algorithm_type pbe_algorithm,
    const char *       password,
    int                iterations,
    gsk_buffer *       data,
    gsk_buffer *       stream)
```

### Parameters

*version*

Specifies the PKCS #7 EncryptedData version number. This must be 0.

*pbe\_algorithm*

Specifies the password-based encryption algorithm.

*password*

Specifies the encryption password as a null-terminated string in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

*iterations*

Specifies the number of iterations used to derive the encryption key from the password. It is recommended that iterations be specified as 1024 or greater.

*data*

Specifies the application data for the EncryptedData message.

*stream*

Returns the ASN.1 DER-encoded stream. The application should call the `gsk_free_buffer()` routine to release the stream when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### [CMSERR\_ALG\_NOT\_AVAILABLE]

Encryption algorithm is not available

#### [CMSERR\_ALG\_NOT\_SUPPORTED]

Encryption algorithm is not supported

#### [CMSERR\_CONTENT\_NOT\_SUPPORTED]

The content type is not supported

#### [CMSERR\_NO\_CONTENT\_DATA]

The content data length is zero

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available

#### [CMSERR\_VERSION\_NOT\_SUPPORTED]

The version is not valid

## **gsk\_make\_encrypted\_data\_msg()**

### **Usage**

The **gsk\_make\_encrypted\_data\_msg()** routine creates a PKCS #7 (Cryptographic Message Syntax) EncryptedData message and returns the ASN.1 DER-encoded ContentInfo sequence. The encrypted data content type will be Data. The **gsk\_read\_encrypted\_data\_msg()** routine can be used to extract the application data from the stream.

Calling the **gsk\_make\_encrypted\_data\_msg()** routine is equivalent to calling the **gsk\_make\_data\_content()** routine, the **gsk\_make\_encrypted\_data\_content()** routine, and the **gsk\_make\_content\_msg()** routine.

The encryption key is derived from the password as described in PKCS #5 (Password-based Encryption) and PKCS #12 (Personal Information Exchange). The selected algorithm determines how the key is derived from the password.

The following password-based encryption algorithms are supported. The strong encryption algorithms may not be available depending upon government export regulations.

- x509\_alg\_pbeWithMd2AndDesCbc - 56-bit DES encryption with MD2 digest - {1.2.840.113549.1.5.1}
- x509\_alg\_pbeWithMd5AndDesCbc - 56-bit DES encryption with MD5 digest - {1.2.840.113549.1.5.3}
- x509\_alg\_pbeWithSha1AndDesCbc - 56-bit DES encryption with SHA-1 digest - {1.2.840.113549.1.5.10}
- x509\_alg\_pbeWithMd2AndRc2Cbc - 64-bit RC2 encryption with MD2 digest - {1.2.840.113549.1.5.4}
- x509\_alg\_pbeWithMd5AndRc2Cbc - 64-bit RC2 encryption with MD5 digest - {1.2.840.113549.1.5.6}
- x509\_alg\_pbeWithSha1AndRc2Cbc - 64-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.5.11}
- x509\_alg\_pbeWithSha1And40BitRc2Cbc - 40-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.6}
- x509\_alg\_pbeWithSha1And128BitRc2Cbc - 128-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.5}
- x509\_alg\_pbeWithSha1And40BitRc4 - 40-bit RC4 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.2}
- x509\_alg\_pbeWithSha1And128BitRc4 - 128-bit RC4 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.1}
- x509\_alg\_pbeWithSha1And3DesCbc - 168-bit 3DES encryption with SHA-1 digest - {1.2.840.113549.1.12.1.3}

## gsk\_make\_enveloped\_data\_content()

Create PKCS #7 EnvelopedData content information

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_enveloped_data_content (
    int                version,
    pkcs_session_key * session_key,
    pkcs_certificates * recipient_certificates,
    pkcs_content_info * content_data,
    pkcs_content_info * content_info)
```

### Parameters

#### *version*

Specifies the PKCS #7 EnvelopedData version number. Specify 0 to create EnvelopedData content as described in PKCS #7 Version 1.5. Specify 1 to create EnvelopedData content as described in PKCS #7 Version 1.6.

#### *session\_key*

Specifies the session encryption key as follows:

- The *encryptionType* field specifies the encryption algorithm.
- The *encryptionKey.length* field specifies the encryption key length in bytes.
- The *encryptionKey.data* field specifies the address of the encryption key. A new key will be generated and returned in this parameter if the key address is NULL. If a new key is generated, the application should call the **gsk\_free\_buffer()** routine to release the key when it is no longer needed. Note that the *encryptionType* and *encryptionKey.length* fields must be set by the application even when a new session key is to be generated.

#### *recipient\_certificates*

Specifies the certificates for the message recipients. There must be at least one recipient.

#### *content\_data*

Specifies the EnvelopedData content. This must be one of the content information types defined in PKCS #7.

#### *content\_info*

Returns the EnvelopedData content information. The application should call the **gsk\_free\_content\_info()** routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_ALG\_NOT\_AVAILABLE]**

The encryption algorithm is not available

#### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The encryption algorithm is not supported

#### **[CMSERR\_BAD\_KEY\_SIZE]**

The encryption key size is not supported

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported

## **gsk\_make\_enveloped\_data\_content()**

### **[CMSERR\_INCORRECT\_KEY\_USAGE]**

A recipient certificate does not allow key encipherment

### **[CMSERR\_KEY\_MISMATCH]**

A recipient public key does not support data encryption

### **[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available

### **[CMSERR\_RECIPIENT\_NOT\_FOUND]**

No recipient certificates provided

### **[CMSERR\_VERSION\_NOT\_SUPPORTED]**

The version is not valid

## **Usage**

The **gsk\_make\_enveloped\_data\_content()** routine creates PKCS #7 (Cryptographic Message Syntax) EnvelopedData content information. The data content type must be one of the types defined by PKCS #7.

The **gsk\_read\_enveloped\_data\_content()** routine can be used to extract the content data from the EnvelopedData content information. No validity checking is performed on the recipient certificates. It is assumed that the application has already validated the recipient certificates.

The session key is used to encrypt the message content. A new session key is generated and returned to the application if no key is provided. For each recipient, the session key is encrypted with the recipient's public key and stored in the EnvelopedData message. This means the public key algorithm must support data encryption. Currently, only RSA public keys support data encryption. In addition, the certificate key usage must allow key encipherment.

The following encryption algorithms are supported. Strong encryption may not be available depending upon government export regulations.

- x509\_alg\_rc2CbcPad - 40-bit and 128-bit RC2 - Key lengths 5 and 16 - {1.2.840.113549.3.2}
- x509\_alg\_rc4 - 40-bit and 128-bit RC4 - Key lengths 5 and 16 - {1.2.840.113549.3.4}
- x509\_alg\_desCbcPad - 56-bit DES - Key length 8 - {1.3.14.3.2.7}
- x509\_alg\_desEde3CbcPad - 168-bit 3DES - Key length 24 - {1.2.840.113549.3.7}

## gsk\_make\_enveloped\_data\_msg()

Creates a PKCS #7 EnvelopedData message from application data.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_enveloped_data_msg (
    int                version,
    pkcs_session_key * session_key,
    pkcs_certificates * recipient_certificates,
    gsk_buffer *      data,
    gsk_buffer *      stream)
```

### Parameters

#### *version*

Specifies the PKCS #7 EnvelopedData version number. Specify 0 to create an EnvelopedData message as described in PKCS #7 Version 1.5. Specify 1 to create an EnvelopedData message as described in PKCS #7 Version 1.6.

#### *session\_key*

Specifies the session encryption key as follows:

- The *encryptionType* field specifies the encryption algorithm.
- The *encryptionKey.length* field specifies the encryption key length in bytes.
- The *encryptionKey.data* field specifies the address of the encryption key. A new key will be generated and returned in this parameter if the key address is NULL. If a new key is generated, the application should call the **gsk\_free\_buffer()** routine to release the key when it is no longer needed. Note that the *encryptionType* and *encryptionKey.length* fields must be set by the application even when a new session key is to be generated.

#### *recipient\_certificates*

Specifies the certificates for the message recipients. There must be at least one recipient.

#### *data*

Specifies the application data for the EnvelopedData message.

#### *stream*

Returns the ASN.1 DER-encoded stream. The application should call the **gsk\_free\_buffer()** routine to release the stream when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_ALG\_NOT\_AVAILABLE]**

The encryption algorithm is not available.

#### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The encryption algorithm is not supported.

#### **[CMSERR\_BAD\_KEY\_SIZE]**

The encryption key size is not supported.

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported.

#### **[CMSERR\_INCORRECT\_KEY\_USAGE]**

A recipient certificate does not allow key encipherment.

## **gsk\_make\_enveloped\_data\_msg()**

### **[CMSERR\_KEY\_MISMATCH]**

A recipient public key does not support data encryption.

### **[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_RECIPIENT\_NOT\_FOUND]**

No recipient certificates provided.

### **[CMSERR\_VERSION\_NOT\_SUPPORTED]**

The version is not valid.

## **Usage**

The **gsk\_make\_enveloped\_data\_msg()** routine creates a PKCS #7 (Cryptographic Message Syntax) EnvelopedData message and returns the ASN.1 DER-encoded ContentInfo sequence. The enveloped data content type will be Data. The **gsk\_read\_enveloped\_data\_msg()** routine can be used to extract the application data from the stream. No validity checking is performed on the recipient certificates. It is assumed that the application has already validated the recipient certificates.

Calling the **gsk\_make\_enveloped\_data\_msg()** routine is equivalent to calling the **gsk\_make\_data\_content()** routine, the **gsk\_make\_enveloped\_data\_content()** routine, and the **gsk\_make\_content\_msg()** routine.

The session key is used to encrypt the message content. A new session key is generated and returned to the application if no key is provided. For each recipient, the session key is encrypted with the recipient's public key and stored in the EnvelopedData message. This means the public key algorithm must support data encryption. Currently, only RSA public keys support data encryption. In addition, the certificate key usage must allow key encipherment.

The following encryption algorithms are supported. Strong encryption may not be available depending upon government export regulations.

- x509\_alg\_rc2CbcPad - 40-bit and 128-bit RC2 - Key lengths 5 and 16 - {1.2.840.113549.3.2}
- x509\_alg\_rc4 - 40-bit and 128-bit RC4 - Key lengths 5 and 16 - {1.2.840.113549.3.4}
- x509\_alg\_desCbcPad - 56-bit DES - Key length 8 - {1.3.14.3.2.7}
- x509\_alg\_desEde3CbcPad - 168-bit 3DES - Key length 24 - {1.2.840.113549.3.7}

## gsk\_make\_signed\_data\_content()

Creates PKCS #7 SignedData content information.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_signed_data_content (
    int                version,
    x509_algorithm_type digest_algorithm,
    gsk_boolean        include_certificates,
    pkcs_cert_keys *   signer_certificates,
    pkcs_certificates * ca_certificates,
    pkcs_content_info * content_data,
    pkcs_content_info * content_info)
```

### Parameters

#### *version*

Specifies the PKCS #7 SignedData version number. Specify 0 to create SignedData content information as described in PKCS #7 Version 1.4, specify 1 to create SignedData content information as described in PKCS #7 Version 1.5, or specify 2 to create SignedData content information as described in PKCS #7 Version 1.6.

#### *digest\_algorithm*

Specifies the digest algorithm.

#### *include\_certificates*

Specify TRUE if the signer and certification authority certificates are to be included in the SignedData content information. Specify FALSE if the certificates are not to be included.

#### *signer\_certificates*

Specifies the certificates and associated private keys for the message signers. There must be at least one signer.

#### *ca\_certificates*

Specifies the certification authority certificates. Zero or more certification authority certificates can be included in the SignedData content information. This parameter is ignored if the `include_certificates` parameter is set to FALSE. NULL can be specified for this parameter if no CA certificates are to be included in the message.

#### *content\_data*

Specifies the SignedData content. This must be one of the content information types defined in PKCS #7.

#### *content\_info*

Returns the SignedData content information. The application should call the `gsk_free_content_info()` routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The digest algorithm is not supported.

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported.

## **gsk\_make\_signed\_data\_content()**

### **[CMSERR\_INCORRECT\_KEY\_USAGE]**

A signer certificate does not allow digital signature.

### **[CMSERR\_KEY\_MISMATCH]**

The digest algorithm is not supported for the private key type.

### **[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_SIGNER\_NOT\_FOUND]**

No signer certificate provided or the certificate is not valid.

### **[CMSERR\_VERSION\_NOT\_SUPPORTED]**

The version is not valid

## **Usage**

The **gsk\_make\_signed\_data\_content()** routine creates PKCS #7 (Cryptographic Message Syntax) SignedData content information. The data content type must be one of the types defined by PKCS #7. The **gsk\_read\_signed\_data\_content()** routine can be used to extract the content data from the SignedData content information. The key usage for the signer certificates must allow digital signature. No validity checking will be performed on the signer certificates. It is assumed that the application has already validated the signer certificates.

A signature is included for each signer provided by the *signer\_certificates* parameter. The X.509 certificates used to sign the message will be included in the SignedData content information if the *include\_certificates* parameter is set to TRUE. The message receiver will need to provide the signer certificates if the *include\_certificates* parameter is set to FALSE.

You can optionally include certification authority certificates in the SignedData content information. These certificate can then be used by the message receiver to validate the signer certificates.

The following digest algorithms are supported.

- x509\_alg\_md2Digest - MD2 digest (RSA keys only) - {1.2.840.113549.2.2}
- x509\_alg\_md5Digest - MD5 digest (RSA keys only) - {1.2.840.113549.2.5}
- x509\_alg\_sha1Digest - SHA-1 digest (RSA and DSA keys) - {1.3.14.3.2.26}

## gsk\_make\_signed\_data\_msg()

Creates a PKCS #7 SignedData message from application data.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_signed_data_msg (
    int                version,
    x509_algorithm_type digest_algorithm,
    gsk_boolean        include_certificates,
    pkcs_cert_keys *   signer_certificates,
    pkcs_certificates * ca_certificates,
    gsk_buffer *        data,
    gsk_buffer *        stream)
```

### Parameters

*version*

Specifies the PKCS #7 SignedData version number. Specify 0 to create a SignedData message as described in PKCS #7 Version 1.4, specify 1 to create a SignedData message as described in PKCS #7 Version 1.5, or specify 2 to create a SignedData message as described in PKCS #7 Version 1.6.

*digest\_algorithm*

Specifies the digest algorithm.

*include\_certificates*

Specify TRUE if the signer and certification authority certificates are to be included in the SignedData message. Specify FALSE if the certificates are not to be included.

*signer\_certificates*

Specifies the certificates and associated private keys for the message signers. There must be at least one signer.

*ca\_certificates*

Specifies the certification authority certificates. Zero or more certification authority certificates can be included in the SignedData message. This parameter is ignored if the `include_certificates` parameter is set to FALSE. NULL can be specified for this parameter if no CA certificates are to be included in the message.

*data*

Specifies the application data for the SignedData message.

*stream*

Returns the ASN.1 DER-encoded stream. The application should call the `gsk_free_buffer()` routine to release the stream when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The digest algorithm is not supported.

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported.

#### **[CMSERR\_INCORRECT\_KEY\_USAGE]**

A signer certificate does not allow digital signature.

## **gsk\_make\_signed\_data\_msg()**

### **[CMSERR\_KEY\_MISMATCH]**

The digest algorithm is not supported for the private key type.

### **[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_SIGNER\_NOT\_FOUND]**

No signer certificate provided or the certificate is not valid.

### **[CMSERR\_VERSION\_NOT\_SUPPORTED]**

The version is not valid.

## **Usage**

The **gsk\_make\_signed\_data\_msg()** routine creates a PKCS #7 (Cryptographic Message Syntax) SignedData message and returns the ASN.1 DER-encoded ContentInfo sequence. The signed data content type will be Data. The **gsk\_read\_signed\_data\_msg()** routine can be used to extract the application data from the stream. The key usage for the signer certificates must allow digital signature. No validity checking will be performed on the signer certificates. It is assumed that the application has already validated the signer certificates.

Calling the **gsk\_make\_signed\_data\_msg()** routine is equivalent to calling the **gsk\_make\_data\_content()** routine, the **gsk\_make\_signed\_data\_content()** routine, and the **gsk\_make\_content\_msg()** routine.

A signature is included for each signer provided by the *signer\_certificates* parameter. The X.509 certificates used to sign the message will be included in the SignedData message if the *include\_certificates* parameter is set to TRUE. The message receiver will need to provide the signer certificates if the *include\_certificates* parameter is set to FALSE.

You can optionally include certification authority certificates in the SignedData message. These certificate can then be used by the message receiver to validate the signer certificates.

The following digest algorithms are supported.

- x509\_alg\_md2Digest - MD2 digest (RSA keys only) - {1.2.840.113549.2.2}
- x509\_alg\_md5Digest - MD5 digest (RSA keys only) - {1.2.840.113549.2.5}
- x509\_alg\_sha1Digest - SHA-1 digest (RSA and DSA keys) - {1.3.14.3.2.26}

## gsk\_make\_wrapped\_content()

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_make_wrapped_content (
                                pkcs_content_info *   content_info,
                                pkcs_content_info *   wrapped_content)
```

### Parameters

*content\_info*

Specifies the content information to be wrapped.

*wrapped\_content*

Returns the wrapped content information. The application should call the **gsk\_free\_content\_info()** routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

**[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported.

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **gsk\_make\_wrapped\_content()** routine wraps the supplied content information in an ASN.1 sequence and returns a new content information containing the wrapped data. The type of the wrapped content information is the same as the type of the original content information. The **gsk\_read\_wrapped\_content()** routine can be used to extract the original content information.

## **gsk\_mktime()**

---

### **gsk\_mktime()**

Converts year/month/day time value to number of seconds since the POSIX epoch

### **Format**

```
#include <gskcms.h>
```

```
gsk_time gsk_mktime (
    gsk_timeval *      ts)
```

### **Parameters**

*ts* Specifies the time to be converted. The *tm\_year*, *tm\_mon*, *tm\_mday*, *tm\_hour*, *tm\_min*, and *tm\_sec* fields are used to generate the converted time.

### **Results**

The return value is the number of seconds since January 1, 1970. Leap seconds are not included in the computation.

### **Usage**

The **gsk\_mktime()** routine converts the time specified in year/month/day format to the number of seconds since the POSIX epoch (January 1, 1970). The **gsk\_mktime()** routine differs from the **mktime()** routine in that the time is UTC and is not adjusted for the local timezone or for daylight savings time.

The year value must be between 1970 and 2106 and is the actual year minus 1900, so *tm\_year* must be between 70 and 206, *tm\_mon* must be between 0 and 11, *tm\_mday* must be between 1 and 31, *tm\_hour* must be between 0 and 23, *tm\_min* must be between 0 and 59, and *tm\_sec* must be between 0 and 59.

---

## gsk\_name\_compare()

Compares two X.509 names.

### Format

```
#include <gskcms.h>
```

```
gsk_boolean gsk_name_compare (  
    x509_name *    name1,  
    x509_name *    name2)
```

### Parameters

*name1*

Specifies the first name to be compared.

*name2*

Specifies the second name to be compare.

### Results

### Usage

The **gsk\_name\_compare()** routine compares two X.509 names and return TRUE if the names are the same and FALSE if they are not the same.

Two names are considered equal if they contain the same sequence of attribute types and attribute values. Attribute values are considered equal if they represent the same character string. If a relative distinguished name (RDN) contains multiple attributes, the attributes must be specified in ascending order based upon their ASN.1 DER encoding. Strings are always stored using UTF-8 encoding.

Printable strings (`gsk_string_printable`) are a special case. Multiple spaces are treated as a single space and the comparison is not case sensitive. Case-sensitive comparisons are used for all other string types.

## `gsk_name_to_dn()`

---

### `gsk_name_to_dn()`

Converts an X.509 name to a DN string.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_name_to_dn (
    x509_name *    name,
    char **       dn)
```

### Parameters

*name*

Specifies the X.509 name to be converted to a distinguished name string. The X.509 strings use UTF-8 encoding.

*dn* Returns the distinguished name in the local code page. The application should call the **gsk\_free\_string()** routine to release the string when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [ASN\_CANT\_CONVERT]

The X.509 name is not a distinguished name.

#### [ASN\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_name\_to\_dn()** routine converts an X.509 name to a distinguished name (DN) string in accordance with RFC 2253 (UTF-8 String Representation of Distinguished Names). The DN string will consist of single-byte characters in the local code page. A double-byte character will be represented using the escaped UTF-8 encoding of the double-byte character in the UCS-2 or UCS-4 character set.

The following DN attribute names are generated by the System SSL runtime. Unrecognized attribute types will be encoded using the numeric object identifier followed by the DER-encoded representation of the attribute value.

- C - Country
- CN - Common name
- DC - Domain component
- EMAIL - E-mail address
- L - Locality
- O - Organization name
- OU - Organizational unit name
- PC - Postal code
- SN - Surname
- ST - State or province
- STREET - Street
- T - Title

## gsk\_open\_database()

Opens a key or request database.

### Format

```
#include <gskcms.h>

gsk_status gsk_open_database (
    const char *      filename,
    const char *      password,
    gsk_boolean       update_mode,
    gsk_handle *      db_handle,
    gskdb_database_type * db_type,
    int *             num_records)
```

### Parameters

#### *filename*

Specifies the database file name in the local code page. The length of the fully-qualified filename cannot exceed 251.

#### *password*

Specifies the database password in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

#### *update\_mode*

Specifies the file access mode. Specify TRUE if the database will be updated and FALSE if the database will not be updated. The application must have write access to the file if TRUE is specified.

#### *db\_handle*

Returns the database handle. The application should call the **gsk\_close\_database()** routine when it no longer needs access to the database.

#### *db\_type*

Returns the database type.

#### *num\_records*

Returns the number of records in the database.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_ACCESS\_DENIED]**

The file permissions do not allow access.

#### **[CMSERR\_BAD\_FILENAME]**

The database file name is not valid.

#### **[CMSERR\_DB\_CORRUPTED]**

The database file is not valid.

#### **[CMSERR\_DB\_LOCKED ]**

The database is open for update by another process.

#### **[CMSERR\_FILE\_NOT\_FOUND]**

The database file is not found.

#### **[CMSERR\_IO\_CANCELED]**

The user canceled the password prompt.

## **gsk\_open\_database()**

### **[CMSERR\_IO\_ERROR ]**

An input/output request failed.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_OPEN\_FAILED]**

Unable to open the database.

## **Usage**

The **gsk\_open\_database()** routine will open a key or request database file for either read-only or read/write access. The database must already exist. The database integrity will be verified and the open will fail if the database has been incorrectly modified. Only one process at a time may open a database in update mode. The database may be accessed by multiple concurrent threads in the same process as long as the same database handle is used by all of the threads.

## gsk\_open\_database\_using\_stash\_file()

Opens a key or request database using a stash file for the database password.

### Format

```
#include <gskcms.h>

gsk_status gsk_open_database_using_stash_file (
    const char *      database_filename,
    const char *      stash_filename,
    gsk_boolean       update_mode,
    gsk_handle *      db_handle,
    gskdb_database_type * db_type,
    int *             num_records)
```

### Parameters

#### *database\_filename*

Specifies the database file name in the local code page. The length of the fully-qualified filename cannot exceed 251.

#### *stash\_filename*

Specifies the stash file name in the local code page. The length of the fully-qualified filename cannot exceed 251. The stash file name always has an extension of ".sth" and the supplied name will be changed if it does not have the correct extension.

#### *update\_mode*

Specifies the file access mode. Specify TRUE if the database will be updated and FALSE if the database will not be updated. The application must have write access to the file if TRUE is specified.

#### *db\_handle*

Returns the database handle. The application should call the **gsk\_close\_database()** routine when it no longer needs access to the database.

#### *db\_type*

Returns the database type.

#### *num\_records*

Returns the number of records in the database.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_ACCESS\_DENIED]**

The file permissions do not allow access.

#### **[CMSERR\_BAD\_FILENAME]**

The database file name is not valid.

#### **[CMSERR\_DB\_CORRUPTED]**

The database file is not valid.

#### **[CMSERR\_DB\_LOCKED ]**

The database is open for update by another process.

#### **[CMSERR\_FILE\_NOT\_FOUND]**

The database file is not found.

#### **[CMSERR\_IO\_ERROR ]**

An input/output request failed.

## **gsk\_open\_database\_using\_stash\_file()**

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_OPEN\_FAILED]**

Unable to open the database.

## **Usage**

The **gsk\_open\_database\_using\_stash\_file()** routine is the same as the **gsk\_open\_database()** routine except the database password is obtained from the password stash file instead of being specified as a call parameter. The key or request database can be opened for read-only access or for read/write access. The database must already exist. The database integrity will be verified and the open will fail if the database has been incorrectly modified. Only one process at a time may open a database in update mode. The database may be accessed by multiple concurrent threads in the same process as long as the same database handle is used by all of the threads.

## gsk\_open\_directory()

Opens an LDAP directory.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_open_directory (
    const char *    server_name,
    int             server_port,
    const char *    user_name,
    const char *    user_password,
    int             crl_cache_timeout,
    gsk_handle *    db_handle)
```

### Parameters

*server\_name*

Specifies one or more blank-separated LDAP server host names. Each host name can contain an optional port number separated from the host name by a colon.

*server\_port*

Specifies the port assigned to the LDAP server. The default port will be used if zero is specified.

*user\_name*

Specifies the distinguished name to be used when binding to the LDAP server. An unauthenticated bind will be done if NULL is specified for this parameter.

*user\_password*

Specifies the password to be used when binding to the LDAP server. NULL may be specified for this parameter when NULL is also specified for the *user\_name* parameter.

*crl\_cache\_timeout*

Specifies the CRL cache timeout interval in hours. Specify 0 to disable CRL caching.

*db\_handle*

Returns the directory handle. The application should call the **gsk\_close\_directory()** routine when it no longer needs access to the LDAP directory.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_LDAP]

Error reported by the LDAP client

#### [CMSERR\_LDAP\_NOT\_AVAILABLE]

LDAP server is not available.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available

### Usage

The **gsk\_open\_directory()** routine will open an LDAP directory and return a directory handle.

## `gsk_open_keyring()`

---

### `gsk_open_keyring()`

Opens a SAF digital certificate key ring.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_open_keyring (
    const char *    ring_name,
    gsk_handle *    db_handle,
    int *           num_records)
```

### Parameters

*ring\_name*

Specifies the ring name in the local code page. When using a key ring owned by the current user, specify the ring name as "name". When using a key ring owned by another user, specify the ring name as "userid/name". The maximum userid length is 8 and the maximum name length is 237.

*db\_handle*

Returns the database handle. The application should call the **`gsk_close_database()`** routine when it no longer needs access to the keyring.

*num\_records*

Returns the number of records in the key ring.

### Results

The function return value will be 0 (**`GSK_OK`**) if no error is detected. Otherwise, it will be one of the return codes listed in the **`gskcms.h`** include file. The following are some common errors:

#### **[CMSERR\_ACCESS\_DENIED]**

The access permissions do not allow access.

#### **[CMSERR\_BAD\_FILENAME]**

The key ring name is not valid.

#### **[CMSERR\_FILE\_NOT\_FOUND]**

The key ring does not exist

#### **[CMSERR\_IO\_ERROR]**

An error occurred while listing the key ring.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **`gsk_open_keyring()`** routine will open a key ring maintained by the System Authorization Facility (SAF) and construct a read-only key database. Only trusted certificates connected to the specified key ring are included in the key database. The **`GSKDB_RECFLAG_DEFAULT`** flag will be set if the certificate is the default certificate for the key ring.

The user must have **`READ`** access to the **`IRR.DIGTCERT.LISTRING`** resource in the **`FACILITY`** class when using a SAF key ring owned by the user. The user must have **`UPDATE`** access to the **`IRR.DIGTCERT.LISTRING`** resource in the **`FACILITY`** class when using a SAF key ring owned by another user. Note that certificate private keys are not available when using a SAF key ring owned by another user.

## **gsk\_query\_crypto\_level()**

Returns the available cryptographic levels.

### **Format**

```
#include <gskcms.h>

void gsk_query_crypto_level (
    int *          cms_version,
    int *          cms_release,
    gsk_uint32 *   crypto_level)
```

### **Parameters**

*cms\_version*  
Returns the runtime version number.

*cms\_release*  
Returns the runtime release number.

*crypto\_level*  
Returns the available cryptographic levels.

### **Results**

The **gsk\_query\_crypto\_level()** routine returns the System SSL runtime version, release, and available cryptographic levels. The current System SSL runtime is Version 3 Release 14. The cryptographic level is a bit mask as follows:

**[GSK\_CRYPT0\_64]**  
Set if 64-bit encryption keys are supported.

**[GSK\_CRYPT0\_128]**  
Set if 128-bit encryption keys are supported.

**[GSK\_CRYPT0\_168]**  
Set if 168-bit encryption keys are supported.

## `gsk_query_database_label()`

---

### `gsk_query_database_label()`

Determines if a database label exists

#### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_query_database_label (
    gsk_handle          db_handle,
    const char *       label)
```

#### Parameters

*db\_handle*

Specifies the database handle returned by the `gsk_create_database()` routine, the `gsk_open_database()` routine, or the `gsk_open_keyring()` routine.

*label*

Specifies the database label. The label is specified in the local code page.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

##### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

##### [CMSERR\_BAD\_LABEL]

No label specified.

##### [CMSERR\_RECORD\_NOT\_FOUND]

The label does not exist in the database.

#### Usage

The `gsk_query_database_label()` routine will check the database for the requested label.

## **gsk\_query\_database\_record\_length()**

Queries the database record length.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_query_database_record_length (
                                     gsk_handle          db_handle,
                                     gsk_size *          record_length)
```

### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine.

*record\_length*

Returns the current database record length. All records in the database have this length.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

### **Usage**

The **gsk\_query\_database\_record\_length()** routine will return the record length for the database. All records in the database have the same length and a database entry cannot span records. The **gsk\_change\_database\_record\_length()** routine can be called to change the database record length.

## **gsk\_rdttime()**

---

### **gsk\_rdttime()**

Converts the number of seconds since the POSIX epoch to year/month/day.

### **Format**

```
#include <gskcms.h>
```

```
gsk_timeval * gsk_rdttime (  
    gsk_time      secs,  
    gsk_timeval * ts)
```

### **Parameters**

*secs*

Specifies the time value to be converted.

*ts* Returns the converted time in the *tm\_year*, *tm\_mon*, *tm\_mday*, *tm\_hour*, *tm\_min*, and *tm\_sec* fields.

### **Usage**

The **gsk\_rdttime()** routine converts the number of seconds since the POSIX epoch (January 1, 1970) to year/month/day format. The year value is the actual year minus 1900 and the month value is the actual month minus 1 (that is, January is 0 and December is 11). The return value is the same as the second parameter (the address of the struct *tm*).

---

## gsk\_read\_content\_msg()

Processes a PKCS #7 message.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_content_msg (  
    gsk_buffer *      stream,  
    pkcs_content_info * content_info)
```

### Parameters

*stream*

Specifies the ASN.1 DER-encoded stream to be processed.

*content\_info*

Returns the content information for the message. The application should call the **gsk\_free\_content\_info()** routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available

### Usage

The **gsk\_read\_content\_msg()** routine processes a PKCS #7 (Cryptographic Message Syntax) content information message and returns the content information. The message content type can be any of the types defined by the PKCS #7 specification.

**gsk\_read\_data\_content()**

---

## **gsk\_read\_data\_content()**

Processes PKCS #7 Data content information.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_read_data_content (
                                pkcs_content_info *  content_info,
                                gsk_buffer *         data)
```

### **Parameters**

*content\_info*

Specifies the content information to be processed.

*data*

Returns the application data. The application should call the **gsk\_free\_buffer()** routine to release the data when it is no longer needed.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not Data.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_read\_data\_content()** routine processes PKCS #7 (Cryptographic Message Syntax) Data content information created by the **gsk\_make\_data\_content()** routine and returns the application data.

---

## gsk\_read\_data\_msg()

Processes a PKCS #7 Data message.

### Format

```
#include <gskcms.h>

gsk_status gsk_read_data_msg (
    gsk_buffer *    stream,
    gsk_buffer *    data)
```

### Parameters

*stream*

Specifies the ASN.1 DER-encoded stream to be processed.

*data*

Returns the application data. The application should call the **gsk\_free\_buffer()** routine to release the data when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_CONTENT\_NOT\_SUPPORTED]

The message content type is not Data.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_read\_data\_msg()** routine processes a PKCS #7 (Cryptographic Message Syntax) Data message created by the **gsk\_make\_data\_msg()** routine and returns the application data. The message content type must be Data.

Calling the **gsk\_read\_data\_msg()** routine is equivalent to calling the **gsk\_read\_content\_msg()** routine followed by the **gsk\_read\_data\_content()** routine.

## `gsk_read_encrypted_data_content()`

---

### `gsk_read_encrypted_data_content()`

Processes PKCS #7 EncryptedData content information.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_encrypted_data_content (
    const char *      password,
    pkcs_content_info * content_info,
    pkcs_content_info * content_data)
```

### Parameters

*password*

Specifies the encryption password as a null-terminated string in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

*content\_info*

Specifies the content information to be processed

*content\_data*

Returns the decrypted content data. The application should call the **`gsk_free_content_info()`** routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**`GSK_OK`**) if no error is detected. Otherwise, it will be one of the return codes listed in the **`gskcms.h`** include file. The following are some common errors:

#### **[CMSERR\_ALG\_NOT\_AVAILABLE ]**

Encryption algorithm is not available.

#### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

Encryption algorithm is not supported.

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The message content type is not EncryptedData or the content of the EncryptedData message is not supported.

#### **[CMSERR\_NO\_CONTENT\_DATA]**

The encrypted data length is zero.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The **`gsk_read_encrypted_data_content()`** routine processes PKCS #7 (Cryptographic Message Syntax) EncryptedData content information created by the **`gsk_make_encrypted_data_content()`** routine and returns the decrypted content data.

The decryption key is derived from the password as described in PKCS #5 (Password-based Encryption) and PKCS #12 (Personal Information Exchange). The selected algorithm determines how the key is derived from the password.

The following password-based encryption algorithms are supported. The strong encryption algorithms may not be available depending upon government export regulations.

- `x509_alg_pbeWithMd2AndDesCbc` - 56-bit DES encryption with MD2 digest - {1.2.840.113549.1.5.1}
- `x509_alg_pbeWithMd5AndDesCbc` - 56-bit DES encryption with MD5 digest - {1.2.840.113549.1.5.3}

## **gsk\_read\_encrypted\_data\_content()**

- | • x509\_alg\_pbeWithSha1AndDesCbc - 56-bit DES encryption with SHA-1 digest - {1.2.840.113549.1.5.10}
- | • x509\_alg\_pbeWithMd2AndRc2Cbc - 64-bit RC2 encryption with MD2 digest - {1.2.840.113549.1.5.4}
- | • x509\_alg\_pbeWithMd5AndRc2Cbc - 64-bit RC2 encryption with MD5 digest - {1.2.840.113549.1.5.6}
- | • x509\_alg\_pbeWithSha1AndRc2Cbc - 64-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.5.11}
- | • x509\_alg\_pbeWithSha1And40BitRc2Cbc - 40-bit RC2 encryption with SHA-1 digest -  
| {1.2.840.113549.1.12.1.6}
- | • x509\_alg\_pbeWithSha1And128BitRc2Cbc - 128-bit RC2 encryption with SHA-1 digest -  
| {1.2.840.113549.1.12.1.5}
- | • x509\_alg\_pbeWithSha1And40BitRc4 - 40-bit RC4 encryption with SHA-1 digest -  
| {1.2.840.113549.1.12.1.2}
- | • x509\_alg\_pbeWithSha1And128BitRc4 - 128-bit RC4 encryption with SHA-1 digest -  
| {1.2.840.113549.1.12.1.1}
- | • x509\_alg\_pbeWithSha1And3DesCbc - 168-bit 3DES encryption with SHA-1 digest -  
| {1.2.840.113549.1.12.1.3}

## `gsk_read_encrypted_data_msg()`

---

### `gsk_read_encrypted_data_msg()`

Processes a PKCS #7 EncryptedData message.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_encrypted_data_msg (  
    const char *      password,  
    gsk_buffer *      stream,  
    gsk_buffer *      data)
```

### Parameters

*password*

Specifies the encryption password as a null-terminated string in the local code page. The user will be prompted to enter the password if NULL is specified for this parameter.

*stream*

Specifies the ASN.1 DER-encoded stream to be processed.

*data*

Returns the decrypted content of the EncryptedData message. The application should call the `gsk_free_buffer()` routine to release the data when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### [CMSERR\_ALG\_NOT\_AVAILABLE]

Encryption algorithm is not available.

#### [CMSERR\_ALG\_NOT\_SUPPORTED]

Encryption algorithm is not supported.

#### [CMSERR\_CONTENT\_NOT\_SUPPORTED]

The message content type is not EncryptedData or the content of the EncryptedData message is not Data.

#### [CMSERR\_NO\_CONTENT\_DATA]

The encrypted data length is zero.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The `gsk_read_encrypted_data_msg()` routine processes a PKCS #7 (Cryptographic Message Syntax) EncryptedData message created by the `gsk_make_encrypted_data_msg()` routine and returns the decrypted message content. The encrypted data content type must be Data.

Calling the `gsk_read_encrypted_data_msg()` routine is equivalent to calling the `gsk_read_content_msg()` routine, the `gsk_read_encrypted_data_content()` routine, and the `gsk_read_data_content()` routine.

The decryption key is derived from the password as described in PKCS #5 (Password-based Encryption) and PKCS #12 (Personal Information Exchange). The selected algorithm determines how the key is derived from the password.

## **gsk\_read\_encrypted\_data\_msg()**

The following password-based encryption algorithms are supported. The strong encryption algorithms may not be available depending upon government export regulations.

- x509\_alg\_pbeWithMd2AndDesCbc - 56-bit DES encryption with MD2 digest - {1.2.840.113549.1.5.1}
- x509\_alg\_pbeWithMd5AndDesCbc - 56-bit DES encryption with MD5 digest - {1.2.840.113549.1.5.3}
- x509\_alg\_pbeWithSha1AndDesCbc - 56-bit DES encryption with SHA-1 digest - {1.2.840.113549.1.5.10}
- x509\_alg\_pbeWithMd2AndRc2Cbc - 64-bit RC2 encryption with MD2 digest - {1.2.840.113549.1.5.4}
- x509\_alg\_pbeWithMd5AndRc2Cbc - 64-bit RC2 encryption with MD5 digest - {1.2.840.113549.1.5.6}
- x509\_alg\_pbeWithSha1AndRc2Cbc - 64-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.5.11}
- x509\_alg\_pbeWithSha1And40BitRc2Cbc - 40-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.6}
- x509\_alg\_pbeWithSha1And128BitRc2Cbc - 128-bit RC2 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.5}
- x509\_alg\_pbeWithSha1And40BitRc4 - 40-bit RC4 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.2}
- x509\_alg\_pbeWithSha1And128BitRc4 - 128-bit RC4 encryption with SHA-1 digest - {1.2.840.113549.1.12.1.1}
- x509\_alg\_pbeWithSha1And3DesCbc - 168-bit 3DES encryption with SHA-1 digest - {1.2.840.113549.1.12.1.3}

`gsk_read_enveloped_data_content()`

---

## `gsk_read_enveloped_data_content()`

Processes PKCS #7 EnvelopedData content information.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_enveloped_data_content (
    pkcs_cert_keys *    recipient_keys,
    pkcs_content_info * content_info,
    x509_algorithm_type * encryption_algorithm,
    gsk_size *          key_size,
    pkcs_content_info * content_data)
```

### Parameters

*recipient\_keys*

Specifies one or more certificates and associated private keys.

*content\_info*

Specifies the content information to be processed.

*encryption\_algorithm*

Returns the encryption algorithm used to encrypt the message content.

*key\_size*

Returns the encryption key size in bytes.

*content\_data*

Returns the EnvelopedData content data. The application should call the `gsk_free_content_info()` routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### [CMSERR\_ALG\_NOT\_AVAILABLE]

The encryption algorithm is not available.

#### [CMSERR\_ALG\_NOT\_SUPPORTED]

The encryption algorithm is not supported.

#### [CMSERR\_BAD\_KEY\_SIZE]

The encryption key size is not supported.

#### [CMSERR\_CONTENT\_NOT\_SUPPORTED]

The message content type is not EnvelopedData or the content of the EnvelopedData message is not supported.

#### [CMSERR\_INCORRECT\_KEY\_USAGE]

The recipient certificate does not allow key encipherment.

#### [CMSERR\_KEY\_MISMATCH]

A recipient private key does not support data decryption.

#### [CMSERR\_NO\_CONTENT\_DATA]

The content data length is zero.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

## **gsk\_read\_enveloped\_data\_content()**

### **[CMSERR\_RECIPIENT\_NOT\_FOUND]**

No matching recipient certificate provided.

### **Usage**

The **gsk\_read\_enveloped\_data\_content()** routine processes PKCS #7 (Cryptographic Message Syntax) EnvelopedData content information created by the **gsk\_make\_enveloped\_data\_content()** routine.

The *recipient\_keys* parameter supplies one or more recipient certificates and associated private keys. The **gsk\_read\_enveloped\_data\_content()** routine will search for a certificate matching one of the message recipients. The private key will be used to decrypt the session key and the session key will then be used to decrypt the enveloped data. The certificate key usage must allow key encipherment.

No certificate validation is performed by the **gsk\_read\_enveloped\_data\_content()** routine. It is assumed that the application has already validated the recipient certificates.

The following encryption algorithms are supported. Strong encryption may not be available depending upon government export regulations.

- x509\_alg\_rc2CbcPad - 40-bit and 128-bit RC2 - {1.2.840.113549.3.2}
- x509\_alg\_rc4 - 40-bit and 128-bit RC4 - {1.2.840.113549.3.4}
- x509\_alg\_desCbcPad - 56-bit DES - {1.3.14.3.2.7}
- x509\_alg\_desEde3CbcPad - 168-bit 3DES - {1.2.840.113549.3.7}

## `gsk_read_enveloped_data_msg()`

---

### `gsk_read_enveloped_data_msg()`

Processes a PKCS #7 EnvelopedData message.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_enveloped_data_msg (
    pkcs_cert_keys *      recipient_keys,
    gsk_buffer *         stream,
    x509_algorithm_type * encryption_algorithm,
    gsk_size *           key_size,
    gsk_buffer *         data)
```

### Parameters

*recipient\_keys*

Specifies one or more certificates and associated private keys.

*stream*

Specifies the ASN.1 DER-encoded stream to be processed.

*encryption\_algorithm*

Returns the encryption algorithm used to encrypt the message content.

*key\_size*

Returns the encryption key size in bytes.

*data*

Returns the content of the EnvelopedData message. The application should call the `gsk_free_buffer()` routine to release the data when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### [CMSERR\_BAD\_ENCODING]

The message content type is not EnvelopedData or the message content is not Data.

#### [CMSERR\_BAD\_KEY\_SIZE]

The encryption key size is not supported.

#### [CMSERR\_CONTENT\_NOT\_SUPPORTED]

The message content type is not EnvelopedData or the content of the EnvelopedData message is not Data.

#### [CMSERR\_INCORRECT\_KEY\_USAGE]

The recipient certificate does not allow key encipherment.

#### [CMSERR\_KEY\_MISMATCH]

A recipient private key does not support data decryption.

#### [CMSERR\_NO\_CONTENT\_DATA]

The content data length is zero.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_RECIPIENT\_NOT\_FOUND]

No matching recipient certificate provided.

## Usage

The **gsk\_read\_enveloped\_data\_msg()** routine processes a PKCS #7 (Cryptographic Message Syntax) EnvelopedData message created by the **gsk\_make\_enveloped\_data\_msg()** routine and returns the message content. The enveloped data content type must be Data.

Calling the **gsk\_read\_enveloped\_data\_msg()** routine is equivalent to calling the **gsk\_read\_content\_msg()** routine, the **gsk\_read\_enveloped\_data\_content()** routine, and the **gsk\_read\_data\_content()** routine.

The `recipient_keys` parameter supplies one or more recipient certificates and associated private keys. The **gsk\_read\_enveloped\_data\_msg()** routine will search for a certificate matching one of the message recipients. The private key will be used to decrypt the session key and the session key will then be used to decrypt the enveloped data. The certificate key usage must allow key encipherment.

No certificate validation is performed by the **gsk\_read\_enveloped\_data\_msg()** routine. It is assumed that the application has already validated the recipient certificates.

The following encryption algorithms are supported. Strong encryption may not be available depending upon government export regulations.

- `x509_alg_rc2CbcPad` - 40-bit and 128-bit RC2 - {1.2.840.113549.3.2}
- `x509_alg_rc4` - 40-bit and 128-bit RC4 - {1.2.840.113549.3.4}
- `x509_alg_desCbcPad` - 56-bit DES - {1.3.14.3.2.7}
- `x509_alg_desEde3CbcPad` - 168-bit 3DES - {1.2.840.113549.3.7}

## `gsk_read_signed_data_content()`

---

### `gsk_read_signed_data_content()`

Processes PKCS #7 SignedData content information.

#### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_signed_data_content (
    pkcs_certificates *    local_certificates,
    pkcs_content_info *    content_info,
    gsk_boolean *         used_local,
    pkcs_certificates *    msg_certificates,
    pkcs_certificates *    signer_certificates,
    pkcs_content_info *    content_data)
```

#### Parameters

*local\_certificates*

Specifies zero or more X.509 certificates to use when verifying the message signatures. NULL can be specified for this parameter if no local certificates are provided.

*content\_info*

Specifies the content information to be processed.

*used\_local*

This parameter will be set to TRUE if the signatures were verified using just the certificates supplied by the *local\_certificates* parameter. This parameter will be set to FALSE if any of the signatures were verified using certificates contained within the message.

*msg\_certificates*

Returns the X.509 certificates contained within the message. The application should call the **gsk\_free\_certificates()** routine to release the certificates when they are no longer needed. Specify NULL for this parameter if the message certificates are not needed.

*signer\_certificates*

Returns the certificates used to sign the message. The application should call the **gsk\_free\_certificates()** routine to release the certificates when they are no longer needed. Specify NULL for this parameter if the signer certificates are not needed.

*content\_data*

Returns the SignedData content data. The application should call the **gsk\_free\_content\_info()** routine to release the data when it is no longer needed.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The digest algorithm is not supported.

##### **[CMSERR\_BAD\_SIGNATURE]**

Signature is not correct.

##### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not SignedData.

##### **[CMSERR\_INCORRECT\_KEY\_USAGE ]**

A signer certificate does not allow digital signature.

##### **[CMSERR\_KEY\_MISMATCH]**

The digest algorithm is not supported for the private key type.

## **gsk\_read\_signed\_data\_content()**

### **[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero.

### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **[CMSERR\_SIGNER\_NOT\_FOUND]**

Signer certificate not found.

## **Usage**

The **gsk\_read\_signed\_data\_content()** routine processes PKCS #7 (Cryptographic Message Syntax) SignedData message created by the **gsk\_make\_signed\_data\_content()** routine and returns the content data.

The *local\_certificates* parameter can supply the signer certificates used to verify the message signatures. If a certificate is not found for a message signer, the **gsk\_read\_signed\_data\_content()** routine will attempt to locate the signer certificate in the SignedData message. An error will be returned if the signer certificate cannot be found or if the certificate key usage does not allow digital signature.

No certificate validation is performed by the **gsk\_read\_signed\_data\_content()** routine. It is assumed that the application has already validated the local certificates. The certificates contained in the SignedData message will be returned in the *msg\_certificates* parameter and the *used\_local* parameter will be set to FALSE if any of these certificates were used to verify the message signatures. It is the responsibility of the application to validate the message certificates (for example, by calling the **gsk\_validate\_certificate()** routine for each of the signer certificates).

The following digest algorithms are supported.

- x509\_alg\_md2Digest - MD2 digest (RSA keys only) - {1.2.840.113549.2.2}
- x509\_alg\_md5Digest - MD5 digest (RSA keys only) - {1.2.840.113549.2.5}
- x509\_alg\_sha1Digest - SHA-1 digest (RSA and DSA keys) - {1.3.14.3.2.26}

## `gsk_read_signed_data_msg()`

---

### `gsk_read_signed_data_msg()`

Processes a PKCS #7 SignedData message.

#### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_signed_data_msg (
    pkcs_certificates *    local_certificates,
    gsk_buffer *          stream,
    gsk_boolean *         used_local,
    pkcs_certificates *    msg_certificates,
    pkcs_certificates *    signer_certificates,
    gsk_buffer *          data)
```

#### Parameters

*local\_certificates*

Specifies zero or more X.509 certificates to use when verifying the message signatures. NULL can be specified for this parameter if no local certificates are provided.

*stream*

Specifies the ASN.1 DER-encoded stream to be processed.

*used\_local*

This parameter will be set to TRUE if the signatures were verified using just the certificates supplied by the *local\_certificates* parameter. This parameter will be set to FALSE if any of the signatures were verified using certificates contained within the message.

*msg\_certificates*

Returns the X.509 certificates contained within the message. The application should call the **gsk\_free\_certificates()** routine to release the certificates when they are no longer needed. Specify NULL for this parameter if the message certificates are not needed.

*signer\_certificates*

Returns the certificates used to sign the message. The application should call the **gsk\_free\_certificates()** routine to release the certificates when they are no longer needed. Specify NULL for this parameter if the signer certificates are not needed.

*data*

Returns the content of the SignedData message. The application should call the **gsk\_free\_buffer()** routine to release the data when it is no longer needed.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[ASN\_NO\_MEMORY]**

Insufficient storage is available.

##### **[ASN\_SELECTION\_OUT\_OF\_RANGE]**

Certificate type or version number is not valid.

##### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The digest algorithm is not supported.

##### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The message content type is not SignedData or the content of the SignedData message is not Data.

**[CMSERR\_BAD\_SIGNATURE]**

Signature is not correct.

**[CMSERR\_INCORRECT\_KEY\_USAGE]**

A signer certificate does not allow digital signature.

**[CMSERR\_KEY\_MISMATCH]**

The digest algorithm is not supported for the private key type.

**[CMSERR\_NO\_CONTENT\_DATA]**

The content data length is zero.

**[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

**[CMSERR\_SIGNER\_NOT\_FOUND]**

Signer certificate not found.

## Usage

The **gsk\_read\_signed\_data\_msg()** routine processes a PKCS #7 (Cryptographic Message Syntax) SignedData message created by the **gsk\_make\_signed\_data\_msg()** routine and returns the message content. The signed data content type must be Data.

Calling the **gsk\_read\_signed\_data\_msg()** routine is equivalent to calling the **gsk\_read\_content\_msg()** routine, the **gsk\_read\_signed\_data\_content()** routine, and the **gsk\_read\_data\_content()** routine.

The *local\_certificates* parameter can supply the signer certificates used to verify the message signatures. If a certificate is not found for a message signer, the **gsk\_read\_signed\_data\_msg()** routine will attempt to locate the signer certificate in the SignedData message. An error will be returned if the signer certificate cannot be found or if the certificate key usage does not allow digital signature.

No certificate validation is performed by the **gsk\_read\_signed\_data\_msg()** routine. It is assumed that the application has already validated the local certificates. The certificates contained in the SignedData message will be returned in the *msg\_certificates* parameter and the *used\_local* parameter will be set to FALSE if any of these certificates were used to verify the message signatures. It is the responsibility of the application to validate the message certificates (for example, by calling the **gsk\_validate\_certificate()** routine for each of the signer certificates).

The following digest algorithms are supported.

- x509\_alg\_md2Digest - MD2 digest (RSA keys only) - {1.2.840.113549.2.2}
- x509\_alg\_md5Digest - MD5 digest (RSA keys only) - {1.2.840.113549.2.5}
- x509\_alg\_sha1Digest - SHA-1 digest (RSA and DSA keys) - {1.3.14.3.2.26}

## `gsk_read_wrapped_content()`

---

### `gsk_read_wrapped_content()`

Processes wrapped content information.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_read_wrapped_content (
    pkcs_content_info *    wrapped_content,
    pkcs_content_info *    content_info)
```

### Parameters

*wrapped\_content*

Specifies the wrapped content information.

*content\_info*

Returns the content information. The application should call the `gsk_free_content_info()` routine to release the content information when it is no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

#### **[CMSERR\_CONTENT\_NOT\_SUPPORTED]**

The content type is not supported.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### Usage

The `gsk_read_wrapped_content()` routine processes an ASN.1 sequence containing encoded content information and returns the unwrapped content information.

## gsk\_receive\_certificate()

Receives one or more certificates.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_receive_certificate (
    gsk_buffer *      stream,
    pkcs_certificates * certificates)
```

### Parameters

*stream*

Specifies the byte stream of the encoded certificate.

*certificate*

Returns the decoded certificates. The application should call the **gsk\_free\_certificates()** routine to release the certificates when they are no longer needed.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BAD\_BASE64\_ENCODING]

The Base64 encoding of the import file is not correct.

#### [CMSERR\_BAD\_ENCODING]

The import file format is not recognized.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_receive\_certificate()** routine receives one or more X.509 certificates and returns the decoded certificates to the caller.

The supplied stream can represent either the ASN.1 DER encoding for the certificate or the Cryptographic Message Syntax (PKCS #7) encoding for the certificate. This can be either the binary value or the Base64 encoding of the binary value. A Base64 encoded stream must be in the local code page and must include the encoding header and footer lines.

A Base64 DER-encoded sequence must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'. A Base 64 PKCS #7 signed data message must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----' or must start with the encoding header '-----BEGIN PKCS #7 SIGNED DATA-----' and end with the encoding footer '-----END PKCS #7 SIGNED DATA-----'.

A DER-encoded certificate stream contains a single X.509 certificate while a PKCS #7 message stream contains one or more certificates. All of the certificates in a PKCS #7 message will be returned to the application for processing.

## **gsk\_replace\_record()**

---

### **gsk\_replace\_record()**

Replaces a record in a key or request database.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_replace_record (
    gsk_handled          db_handle,
    gskdb_record *      record)
```

### **Parameters**

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine.

*record*

Specifies the database record.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BACKUP\_EXISTS]**

The backup file already exists.

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

#### **[CMSERR\_BAD\_LABEL]**

The record label is not valid.

#### **[CMSERR\_DEFAULT\_KEY\_CHANGED]**

The default key cannot be changed.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The record type is not supported for the database type.

#### **[CMSERR\_IO\_ERROR]**

Unable to write record.

#### **[CMSERR\_LABEL\_NOT\_UNIQUE]**

The record label is not unique.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

#### **[CMSERR\_NO\_PRIVATE\_KEY]**

No private key is provided for a record type that requires a private key.

#### **[CMSERR\_PUBLIC\_KEY\_CHANGED]**

The subject public key cannot be changed.

#### **[CMSERR\_RECORD\_NOT\_FOUND]**

Record is not found.

#### **[CMSERR\_RECORD\_TOO\_BIG]**

The record is larger than the database record length.

#### **[CMSERR\_RECTYPE\_NOT\_VALID]**

The record type is not valid.

**[CMSERR\_SUBJECT\_CHANGED]**

The subject name cannot be changed.

**[CMSERR\_UPDATE\_NOT\_ALLOWED]**

Database is not open for update.

## Usage

The **gsk\_replace\_record()** routine replaces a record in a key or request database. The database must be open for update in order to replace records. The unique record identifier identifies the record to be replaced. Unused and reserved fields in the `gskdb_record` structure must be initialized to zero. If the record has a private key, the encrypted private key will be generated from the private key supplied in the database record.

The `recordType` field identifies the database record type as follows:

**gskdb\_rectype\_certificate**

The record contains an X.509 certificate.

**gskdb\_rectype\_certKey**

The record contains an X.509 certificate and private key.

**gskdb\_rectype\_keyPairTerm**

The record contains a PKCS #10 certification request and private key.

The `recordFlags` field is a bit field with the following values:

**GSKDB\_RECFLAG\_TRUSTED**

The certificate is trusted.

**GSKDB\_RECFLAG\_DEFAULT**

This is the default key

The record label is used as a friendly name for the database entry and is in the local code page. It can be set to any value and consists of characters which can be represented using 7-bit ASCII (letters, numbers, and punctuation). It may not be set to an empty string.

If the record contains a certificate, the certificate will be validated and the record will not be replaced in the database if the validation check fails.

With the exception of the record label, all character strings are specified using UTF-8.

The record type, subject name, and subject public key cannot be changed when replacing a record. In addition, the `GSKDB_RECFLAG_DEFAULT` flag cannot be changed when replacing a record (call the **gsk\_set\_default\_key()** routine to change the default record for the database).

The database file is updated as part of the **gsk\_replace\_record()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## gsk\_set\_default\_key()

---

### gsk\_set\_default\_key()

Sets the default key.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_set_default_key (
                                gsk_handle      db_handle,
                                gsk_int32       record_id)
```

### Parameters

*db\_handle*

Specifies the database handle returned by the **gsk\_create\_database()** routine or the **gsk\_open\_database()** routine.

*record\_id*

Specifies the unique record identifier of the new default key.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_BACKUP\_EXISTS]

The backup file already exists.

#### [CMSERR\_BAD\_HANDLE]

The database handle is not valid.

#### [CMSERR\_INCORRECT\_DBTYPE]

The database type does not support a default key.

#### [CMSERR\_IO\_ERROR]

Unable to write record.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

#### [CMSERR\_NO\_PRIVATE\_KEY]

The database record does not contain a private key.

#### [CMSERR\_RECORD\_NOT\_FOUND]

Record is not found.

#### [CMSERR\_UPDATE\_NOT\_ALLOWED]

Database is not open for update.

### Usage

The **gsk\_set\_default\_key()** routine sets the default key for a key database. If the key database already has a default key, the record for the old default key is updated to remove the **GSKDB\_RECFLAG\_DEFAULT** flag. The record for the new default key is then updated to add the **GSKDB\_RECFLAG\_DEFAULT** flag. The database must be open for update in order to set the default key. An error will be returned if the specified database record does not contain a private key.

The database file is updated as part of the **gsk\_replace\_record()** processing. A temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then overwritten and the temporary database file is deleted. The temporary database file will not be deleted if an error occurs while rewriting the database file.

## gsk\_sign\_certificate()

Signs an X.509 certificate.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_sign_certificate (
    x509_certificate *      certificate,
    pkcs_private_key_info * private_key)
```

### Parameters

*certificate*

Specifies the X.509 certificate.

*private\_key*

Specifies the private key.

### Results

The return status will be zero if the signature is successfully generated. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_ALG\_NOT\_SUPPORTED]

The signature algorithm is not supported.

#### [CMSERR\_KEY\_MISMATCH]

The supplied key does not match the signature algorithm.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_sign\_certificate()** routine will sign an X.509 certificate using the supplied private key. The private key can be an RSA key or a DSA key. The private key can be an ASN.1-encoded value contained in the `privateKey` field or an ICSF key label contained in the `keyToken` field. In either case, the key type must be specified by the `privateKeyAlgorithm` field.

The signature algorithm is obtained from the `signature` field of the `x509_tbs_certificate` structure contained within the `x509_certificate` structure. The generated signature will be placed in the `signatureAlgorithm` and `signatureValue` fields of the `x509_certificate` structure.

The following signature algorithms are supported:

- `x509_alg_md2WithRsaEncryption` - RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}
- `x509_alg_md5WithRsaEncryption` - RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}
- `x509_alg_sha1WithRsaEncryption` - RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}
- `x509_alg_dsaWithSha1` - Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}

## **gsk\_sign\_crl()**

---

### **gsk\_sign\_crl()**

Signs an X.509 certificate revocation list.

### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_sign_crl (
    x509_crl *          crl,
    pkcs_private_key_info * private_key)
```

### **Parameters**

*crl* Specifies the X.509 certificate revocation list.

*private\_key*  
Specifies the private key.

### **Results**

The return status will be zero if the signature is successfully generated. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The signature algorithm is not supported.

#### **[CMSERR\_KEY\_MISMATCH]**

The supplied key does not match the signature algorithm.

#### **[CMSERR\_NO\_MEMORY]**

Insufficient storage is available.

### **Usage**

The **gsk\_sign\_crl()** routine will sign an X.509 certificate revocation list using the supplied private key. The private key can be an RSA key or a DSA key. The private key can be an ASN.1-encoded value contained in the `privateKey` field or an ICSF key label contained in the `keyToken` field. In either case, the key type must be specified by the `privateKeyAlgorithm` field.

The signature algorithm is obtained from the `signature` field of the `x509_tbs_crl` structure contained within the `x509_crl` structure. The generated signature will be placed in the `signatureAlgorithm` and `signatureValue` fields of the `x509_crl` structure.

The following signature algorithms are supported:

- `x509_alg_md2WithRsaEncryption` - RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}
- `x509_alg_md5WithRsaEncryption` - RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}
- `x509_alg_sha1WithRsaEncryption` - RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}
- `x509_alg_dsaWithSha1` - Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}

## gsk\_sign\_data()

Signs a data stream.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_sign_data (
    x509_algorithm_type      sign_algorithm,
    pkcs_private_key_info *  private_key,
    gsk_boolean              is_digest,
    gsk_buffer *             data,
    gsk_buffer *             signature)
```

### Parameters

*sign\_algorithm*

Specifies the signature algorithm.

*private\_key*

Specifies the private key.

*is\_digest*

Specify TRUE if the data stream digest has been computed or FALSE if the data stream digest needs to be computed.

*data*

Specifies either the data stream digest (*is\_digest* is TRUE) or the data stream (*is\_digest* is FALSE).

*signature*

Returns the generated signature. The caller should release the signature buffer when it is no longer needed by calling the **gsk\_free\_buffer()** routine.

### Results

The return status will be zero if the signature is successfully generated. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_ALG\_NOT\_SUPPORTED]

The signature algorithm is not supported.

#### [CMSERR\_BAD\_DIGEST\_SIZE ]

The digest size is not correct.

#### [CMSERR\_KEY\_MISMATCH]

The supplied key does not match the signature algorithm.

#### [CMSERR\_NO\_MEMORY]

Insufficient storage is available.

### Usage

The **gsk\_sign\_data()** routine will generate the signature for a data stream using the supplied private key. The private key can be an RSA key or a DSA key. The private key can be an ASN.1-encoded value contained in the *privateKey* field or an ICSF key label contained in the *keyToken* field. In either case, the key type must be specified by the *privateKeyAlgorithm* field.

The application can either provide the message digest or have the **gsk\_sign\_data()** routine compute the message digest.

When the application provides the message digest, the digest length must be correct for the specified signature algorithm. MD2 and MD5 digests are 16 bytes while a SHA-1 digest is 20 bytes. The supplied

## **gsk\_sign\_data()**

| digest will be used as-is without any further processing (specifically, for an RSA encryption key, the digest  
| will not be encoded as an ASN.1 DigestInfo sequence before generating the signature)

| The following signature algorithms are supported:

- | • x509\_alg\_md2WithRsaEncryption - RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}
- | • x509\_alg\_md5WithRsaEncryption - RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}
- | • x509\_alg\_sha1WithRsaEncryption - RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}
- | • x509\_alg\_dsaWithSha1 - Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}
- | • x509\_alg\_md5Sha1WithRsaEncryption - RSA encryption with combined MD5 and SHA-1 digests

## gsk\_validate\_certificate()

Validates an X.509 certificate.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_validate_certificate (
    gskdb_data_sources *    data_sources,
    x509_certificate *     subject_certificate,
    gsk_boolean            accept_root,
    gsk_int32 *            issuer_record_id)
```

### Parameters

#### *data\_sources*

Specifies the data sources for CA certificates and revocation lists. The data sources are searched in the order they occur in the data source array, so trusted sources should be included before untrusted sources and local sources should be included before remote sources.

#### *subject\_certificate*

Specifies the certificate to be validated.

#### *accept\_root*

Specify TRUE if a self-signed root certificate is to be accepted without checking the data sources. Specify FALSE if a self-signed root certificate must be found in one of the trusted data sources in order to be accepted.

#### *issuer\_record\_id*

Returns the record identifier for the issuer certificate used to validate the certificate. The record identifier will be 0 if the issuer certificate is found in a non-database source. Specify NULL for this parameter if the issuer record identifier is not needed.

### Results

The return status will be zero if the validation is successful. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### **[CMSERR\_BAD\_HANDLE]**

The database handle is not valid.

#### **[CMSERR\_BAD\_ISSUER\_NAME]**

The certificate issuer name is not valid.

#### **[CMSERR\_BAD\_SIGNATURE]**

The signature is not correct.

#### **[CMSERR\_CERT\_CHAIN\_NOT\_TRUST]**

The certification chain is not trusted

#### **[CMSERR\_CERTIFICATE\_REVOKED]**

The certificate is revoked.

#### **[CMSERR\_EXPIRED]**

The certificate is expired.

#### **[CMSERR\_INCORRECT\_DBTYPE]**

The database type does not support certificates.

#### **[CMSERR\_INCORRECT\_KEY\_USAGE]**

The issuer certificate does not allow signing certificates

## **gsk\_validate\_certificate()**

### **[CMSERR\_ISSUER\_NOT\_CA]**

The certificate issuer is not a certification authority.

### **[CMSERR\_ISSUER\_NOT\_FOUND]**

The issuer certificate is not found in one of the data sources.

### **[CMSERR\_NAME\_CONSTRAINTS\_VIOLATED]**

The certificate name is not consistent with the name constraints.

### **[CMSERR\_NAME\_NOT\_SUPPORTED]**

The AuthorityKeyIdentifier extension name is not a directory name.

### **[CMSERR\_NOT\_YET\_VALID]**

The certificate is not yet valid.

### **[CMSERR\_PATH\_TOO\_LONG]**

The certification chain exceeds the maximum allowed by the CA.

### **[CMSERR\_SELF\_SIGNED\_NOT\_FOUND]**

A self-signed certificate is not found in a trusted data source

## **Usage**

The **gsk\_validate\_certificate()** routine validates an X.509 certificate by performing the following checks on the subject certificate:

- The certificate subject name must be either a non-empty distinguished name or an empty distinguished name with a SubjectAltName certificate extension
- An empty subject name is not allowed for a CA certificate
- The certificate issuer name must not be an empty distinguished name
- The CertificatePolicy extension, if present, must not be a critical extension
- The current time must not be earlier than the start of the certificate validity period
- The current time must not be later than the end of the certificate validity period
- The issuer certificate must be a valid CA certificate
- The certificate signature must be correct
- The certificate must not be revoked
- The certification chain must lead to a certificate obtained from a trusted data source
- No certificate in the certification chain can be revoked or expired.

The **gsk\_validate\_certificate()** routine will obtain any necessary CA certificates from the supplied data sources. The CA certificate will be validated as described above if it is obtained from an untrusted data source. In addition, the following checks will be performed on CA certificates when validating the certification chain:

- The BasicConstraints extension, if present, must have the CA indicator set and the path length constraint must not be violated by subordinate certificates in the certification chain
- The NameConstraints extension, if present, must not be violated by the subject certificate

A root certificate is a self-signed certificate and its signature is verified using the public key in the certificate. If `accept_root` is `FALSE`, the root certificate must be found in a trusted data source in order to be accepted. If `accept_root` is `TRUE`, the self-signed certificate is accepted as long as the signature is correct.

An intermediate certificate or an end-entity certificate is a certificate signed by another entity. Its signature is verified using the public key in the issuer's certificate. The issuer certificate must be found in one of the supplied data sources. When intermediate CA certificates are used, the certificate chain is validated until an issuer is reached whose certificate is in one of the trusted data sources.

## **gsk\_validate\_certificate()**

The data sources must contain at least one LDAP directory source or CRL source in order to check for revoked certificates. The CRL distribution point name (or the certificate issuer name if the certificate does not have a CrlDistributionPoints extension) is used as the distinguished name of the LDAP directory entry containing the certificate revocation list (CRL). The CRL distribution point name and CRL issuer name must be X.500 directory names. The BasicConstraints certificate extension determines whether the CA revocation list or the user revocation list is used. An error will be returned if a CRL obtained from an untrusted source cannot be validated.

The following data sources are supported:

- `gskdb_source_key_database` - The source is a key database. The handle must be a database handle returned by the **gsk\_create\_database()** routine, the **gsk\_open\_database()** routine, or the **gsk\_open\_keyring()** routine. This is a trusted data source.
- `gskdb_source_directory` - The source is an LDAP directory. The handle must be the directory handle returned by the **gsk\_open\_directory()** routine. This is an untrusted data source. Any certificate or revocation list obtained from this source will be validated before being accepted. Refer to the **gsk\_get\_directory\_certificates()** and **gsk\_get\_directory\_crls()** routines for more information concerning the use of LDAP directory entries.
- `gskdb_source_trusted_certs` - The source is an array of certificates. This is a trusted data source.
- `gskdb_source_untrusted_certs` - The source is an array of certificates. This is an untrusted data source. Any certificate used from this list will be validated before being accepted.
- `gskdb_source_trusted_crls` - The source is an array of certificate revocation lists. This is a trusted data source.
- `gskdb_source_untrusted_crls` - The source is an array of certificate revocation lists. This is an untrusted data source. Any CRL used from this list will be validated before being accepted.
- `gskdb_source_cert_callback` - The source is the address of a callback routine which will receive control when an issuer certificate is needed. This is a trusted data source. The subject name is passed as an input parameter and the `certCallback` routine returns an array of one or more certificates with that subject name. The **gsk\_validate\_certificate()** routine will call the `freeCallback` routine to release the certificates. The return status should be 0 if no errors are detected. Otherwise it should be one of the error code listed in the **gskcms.h** include file. The return status should be 0 and the certificate count should be 0 if there are no certificates matching the supplied subject name.
- `gskdb_source_crl_callback` - The source is the address of a callback routine which will receive control when a certificate needs to be checked to see if it has been revoked. The return value should be 0 if the certificate is not revoked. Otherwise it should be one of the error codes defined in the **gskcms.h** include file.

## **gsk\_verify\_certificate\_signature()**

---

### **gsk\_verify\_certificate\_signature()**

Verifies the signature for an X.509 certificate.

#### **Format**

```
#include <gskcms.h>
```

```
gsk_status gsk_verify_certificate_signature (
    x509_certificate *    certificate,
    x509_public_key_info * key)
```

#### **Parameters**

*certificate*

Specifies the decoded certificate returned by the **gsk\_decode\_certificate()** routine.

*key*

Specifies the public key for the Certification Authority that signed the certificate.

#### **Results**

The return status will be zero if the signature is correct. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

##### **[CMSERR\_ALG\_NOT\_SUPPORTED]**

The signature algorithm is not supported.

##### **[CMSERR\_BAD\_SIGNATURE]**

The signature is not correct.

##### **[CMSERR\_KEY\_MISMATCH]**

The supplied key does not match the signature algorithm.

#### **Usage**

The **gsk\_verify\_certificate\_signature()** routine validates an X.509 certificate by computing its signature and then comparing the result to the signature contained in the certificate.

The following signature algorithms are supported:

- x509\_alg\_md2WithRsaEncryption - RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}
- x509\_alg\_md5WithRsaEncryption - RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}
- x509\_alg\_sha1WithRsaEncryption - RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}
- x509\_alg\_dsaWithSha1 - Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}

## gsk\_verify\_crl\_signature()

Verifies the signature for an X.509 certificate revocation list.

### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_verify_crl_signature (
    x509_crl *          crl,
    x509_public_key_info * key)
```

### Parameters

*crl* Specifies the decoded certificate revocation list returned by the **gsk\_decode\_crl()** routine.

*key*

Specifies the public key for the Certification Authority that signed the certificate revocation list.

### Results

The return status will be zero if the signature is correct. Otherwise, it will be one of the return codes listed in the **gskcms.h** include file. The following are some common errors:

#### [CMSERR\_ALG\_NOT\_SUPPORTED]

The signature algorithm is not supported.

#### [CMSERR\_BAD\_SIGNATURE]

The signature is not correct.

#### [CMSERR\_KEY\_MISMATCH]

The supplied key does not match the signature algorithm.

### Usage

The **gsk\_verify\_crl\_signature()** routine validates an X.509 certificate revocation list (CRL) by computing its signature and then comparing the result to the signature contained in the CRL.

The following signature algorithms are supported:

- x509\_alg\_md2WithRsaEncryption - RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}
- x509\_alg\_md5WithRsaEncryption - RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}
- x509\_alg\_sha1WithRsaEncryption - RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}
- x509\_alg\_dsaWithSha1 - Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}

## `gsk_verify_data_signature()`

---

### `gsk_verify_data_signature()`

Verifies the signature for a data stream.

#### Format

```
#include <gskcms.h>
```

```
gsk_status gsk_verify_data_signature (
    x509_algorithm_type      sign_algorithm,
    x509_public_key_info *   key,
    gsk_boolean              is_digest,
    gsk_buffer *             data,
    gsk_buffer *             signature)
```

#### Parameters

*sign\_algorithm*

Specifies the signature algorithm.

*key*

Specifies the public key.

*is\_digest*

Specify TRUE if the data stream digest has been computed or FALSE if the data stream digest needs to be computed.

*data*

Specifies either the data stream digest (*is\_digest* is TRUE) or the data stream (*is\_digest* is FALSE).

*signature*

Specifies the data stream signature.

#### Results

The return status will be zero if the signature is correct. Otherwise, it will be one of the return codes listed in the `gskcms.h` include file. The following are some common errors:

##### [CMSERR\_ALG\_NOT\_SUPPORTED]

The signature algorithm is not supported.

##### [CMSERR\_BAD\_DIGEST\_SIZE]

The digest size is not correct.

##### [CMSERR\_BAD\_SIGNATURE]

The signature is not correct.

##### [CMSERR\_KEY\_MISMATCH]

The supplied key does not match the signature algorithm.

#### Usage

The `gsk_verify_data_signature()` routine validates the signature for a data stream. The public key can be an RSA key or a DSA key.

The application can either provide the message digest or have the `gsk_verify_signed_data()` routine compute the message digest.

When the application provides the message digest, the digest length must be correct for the specified signature algorithm. MD2 and MD5 digests are 16 bytes while a SHA-1 digest is 20 bytes. The supplied digest will be used as-is without any further processing (specifically, for an RSA encryption key, the digest will not be encoded as an ASN.1 DigestInfo sequence before comparing it with the digest in the signature)

## **gsk\_verify\_data\_signature()**

| The following signature algorithms are supported:

- | • x509\_alg\_md2WithRsaEncryption - RSA encryption with MD2 digest - {1.2.840.113549.1.1.2}
- | • x509\_alg\_md5WithRsaEncryption - RSA encryption with MD5 digest - {1.2.840.113549.1.1.4}
- | • x509\_alg\_sha1WithRsaEncryption - RSA encryption with SHA-1 digest - {1.2.840.113549.1.1.5}
- | • x509\_alg\_dsaWithSha1 - Digital Signature Standard with SHA-1 digest - {1.2.840.10040.4.3}
- | • x509\_alg\_md5Sha1WithRsaEncryption - RSA encryption with combined MD5 and SHA-1 digests

| The x509\_alg\_md5Sha1WithRsaEncryption algorithm is a special algorithm used by the SSL protocol. The data signature consists of the MD5 digest over the data followed by the SHA-1 digest over the data for a total digest length of 36 bytes. The digest is encrypted as-is without any further processing.

`gsk_verify_data_signature()`

---

## Chapter 9. Deprecated Secure Sockets Layer APIs

The following set of application program interfaces, or APIs, have been superseded by the APIs defined in Chapter 7, “API Reference” on page 27.

- **gsk\_free\_memory()** (see page 226)
- **gsk\_get\_cipher\_info()** (see page 227)
- **gsk\_get\_dn\_by\_label()** (see page 228 )
- **gsk\_initialize()** (see page 229)
- **gsk\_secure\_soc\_close()** (see page 232)
- **gsk\_secure\_soc\_init()** (see page 233)
- **gsk\_secure\_soc\_read()** (see page 238)
- **gsk\_secure\_soc\_reset()** (see page 240)
- **gsk\_secure\_soc\_write()** (see page 241)
- **gsk\_srb\_initialize()** (see page 243)
- **GSKSRBRD()** (see page 244)
- **GSKSRBWT()** (see page 245)
- **gsk\_uninitialize()** (see page 246)
- **gsk\_user\_set()** (see page 247)

Although use of the deprecated set of APIs in this chapter is still supported in z/OS Version 1 Release 4, it is **strongly recommended** that new applications be developed using the set of APIs defined in Chapter 7, “API Reference”.

In addition, it is **strongly recommended** that existing applications be modified to make use of the set of APIs defined in Chapter 7, “API Reference”. Those modified applications should **only** use the new APIs, and **not** a mix of the new APIs and these deprecated APIs. Information about modifying your existing application programs to use the new API set can be found in Chapter 5, “Migrating to the New SSL Interfaces” on page 23.

**IBM may remove support of APIs contained within this chapter in a future release.**

## **gsk\_free\_memory()**

---

### **gsk\_free\_memory()**

Releases storage allocated by the SSL runtime.

#### **Format**

```
#include <gskssl.h>

void gsk_free_memory(
    void *    address,
    void *    reserved)
```

#### **Parameters**

##### **address**

Specifies the address of the storage to be released.

##### **reserved**

Reserved for future use. Specify NULL for this parameter.

#### **Usage**

The **gsk\_free\_memory()** routine releases storage allocated by the SSL runtime.

#### **Related Topics**

**gsk\_get\_dn\_by\_label()**

## gsk\_get\_cipher\_info()

Returns the supported cipher specifications.

### Format

```
#include <gskssl.h>

gsk_status gsk_get_cipher_info(
    int                level,
    gsk_sec_level *    sec_level,
    void *             rsvd)
```

### Parameters

#### level

Specifies GSK\_LOW\_SECURITY to return just the export cipher specifications or GSK\_HIGH\_SECURITY to return the domestic cipher specifications as well as the export cipher specifications.

#### sec\_level

Returns the cipher specifications.

#### rsvd

Reserved for future use. Specify NULL for this parameter.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following is a common error:

#### [GSK\_BAD\_PARAMETER]

The level value is not valid or a NULL address is specified for sec\_level.

### Usage

The **gsk\_get\_cipher\_info()** routine returns the available cipher specifications. Both domestic and export ciphers will be included if GSK\_HIGH\_SECURITY is specified while only export ciphers will be included if GSK\_LOW\_SECURITY is specified. The **gsk\_get\_cipher\_info()** routine can be called at any time and does not require the **gsk\_initialize()** routine to be called first.

The SSL V2 cipher specifications returned for GSK\_HIGH\_SECURITY are "713642" while the SSL V3 cipher specifications are "0504352F0A090306020100". If the Security Level 3 FMID is not installed, the SSL V2 cipher specifications are "642" while the SSL V3 cipher specifications are "090306020100".

The SSL V2 cipher specifications returned for GSK\_LOW\_SECURITY are "642" while the SSL V3 cipher specifications are "090306020100".

### Related Topics

**gsk\_secure\_soc\_init()**

**gsk\_initialize()**

## **gsk\_get\_dn\_by\_label()**

---

### **gsk\_get\_dn\_by\_label()**

Gets the distinguished name for a certificate.

#### **Format**

```
#include <gskssl.h>

char * gsk_get_dn_by_label(
    const char * label)
```

#### **Parameters**

##### **label**

Specifies the key label.

#### **Usage**

The **gsk\_get\_dn\_by\_label()** routine returns the distinguished name for the certificate associated with the key label. The **gsk\_initialize()** routine must be called before the **gsk\_get\_dn\_by\_label()** routine can be called. The application should release the returned name when it is no longer needed by calling the **gsk\_free\_memory()** routine. The return value will be NULL if an error occurred while accessing the key database.

#### **Related Topics**

**gsk\_initialize()**

**gsk\_secure\_soc\_init()**

**gsk\_free\_memory()**

## gsk\_initialize()

Initializes the System SSL runtime environment.

### Format

```
#include <gskssl.h>

gsk_status gsk_initialize(
    gsk_init_data *    init_data)
```

### Parameters

#### **init\_data**

Specifies the data used to initialize the SSL runtime environment.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### **[GSK\_ERR\_INIT\_PARM\_NOT\_VALID]**

An initialization parameter is not valid.

#### **[GSK\_ERROR\_BAD\_MALLOC]**

Insufficient storage is available.

#### **[GSK\_ERROR\_CRYPTO]**

Cryptographic error detected.

#### **[GSK\_ERROR\_LDAP]**

Unable to initialize the LDAP client.

#### **[GSK\_ERROR\_PERMISSION\_DENIED]**

Not authorized to access the key database or key ring.

#### **[GSK\_INIT\_SEC\_TYPE\_NOT\_VALID]**

The security type is not valid.

#### **[GSK\_INIT\_V2\_TIMEOUT\_NOT\_VALID]**

The SSL V2 timeout is not valid.

#### **[GSK\_INIT\_V3\_TIMEOUT\_NOT\_VALID]**

The SSL V3 timeout is not valid.

#### **[GSK\_KEYFILE\_BAD\_FORMAT]**

Key database or key ring format is not valid.

#### **[GSK\_KEYFILE\_BAD\_PASSWORD]**

Key database password is not correct.

#### **[GSK\_KEYFILE\_IO\_ERROR]**

Unable to read the key database or key ring.

#### **[GSK\_KEYFILE\_NO\_CERTIFICATES]**

The key database or key ring does not contain any certificates.

#### **[GSK\_KEYFILE\_OPEN\_FAILED]**

Unable to open the key database or key ring.

#### **[GSK\_KEYFILE\_PW\_EXPIRED]**

Key database password is expired.

## **gsk\_initialize()**

### **Usage**

The **gsk\_initialize()** routine initializes the System SSL runtime environment for the current process. The **gsk\_initialize()** routine is not needed if the application does not use any of the deprecated SSL API routines. The **gsk\_uninitialize()** routine should be called to release the SSL environment when it is no longer needed. Multiple calls to **gsk\_initialize()** will cause the existing environment to be released before creating the new environment.

Environment variables are processed and the key database or key ring is read as part of the environment initialization. Upon successful completion of **gsk\_initialize()**, the application is ready to begin creating and using secure socket connections.

The `gsk_init_data` structure contains the following fields:

#### *sec\_types*

Specifies one of the following null-terminated character strings: "SSLV2" or "SSL20" to use the SSL V2 protocol, "SSLV3" or "SSL30" to use the SSL V3 protocol, "TLSV1" or "TLS10" to use the TLS V1 protocol, or "ALL" to use any supported protocol.

When "ALL" is specified for an SSL client, the client will attempt to use the TLS V1 protocol and will fall-back to the SSL V3 protocol if the SSL server does not support the TLS V1 protocol. No attempt will be made to use the SSL V2 protocol (the client must explicitly request the SSL V2 protocol if it wants to use this protocol).

When "ALL" is specified for an SSL server, the server will accept any of the supported protocols.

#### *keyring*

Specifies the name of the key database or SAF key ring as a null-terminated character string. A SAF key ring is used if both the password and stash file name are NULL.

The SAF key ring name is specified as "userid/keyring". The current userid is used if the userid is omitted. The user must have READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by the user. The user must have UPDATE access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by another user. Note that certificate private keys are not available when using a SAF key ring owned by another user.

#### *keyring\_pw*

Specifies the password for the key database as a null-terminated character string. Specify NULL to indicate no password is provided.

#### *keyring\_stash*

Specifies the name of the password stash file as a null-terminated character string. Specify NULL to indicate no stash file is provided. The password stash file is used if the `keyring_pw` value is NULL.

#### *V2\_session\_timeout*

Specifies the SSL V2 session cache timeout value in seconds. The valid range is 0 to 100. A short SSL handshake is performed when a cached session exists since the session parameters have already been negotiated between the client and the server.

#### *V3\_session\_timeout*

Specifies the SSL V3 session cache timeout value in seconds. The valid range is 0 to 86400. A short SSL handshake is performed when a cached session exists since the session parameters have already been negotiated between the client and the server.

#### *LDAP\_server*

Specifies one or more blank-separated LDAP server host names as a null-terminated character string. Each host name can contain an optional port number separated from the host name by a colon. The LDAP server is used for certificate validation. The LDAP server is used only when

## **gsk\_initialize()**

LDAP\_CA\_roots is set to GSK\_CA\_ROOTS\_LOCAL\_AND\_X500 and *auth\_type* is not set to GSK\_CLIENT\_AUTH\_LOCAL or GSK\_CLIENT\_AUTH\_PASSTHRU.

### *LDAP\_port*

Specifies the LDAP server port. The default LDAP port will be used if 0 is specified.

### *LDAP\_user*

Specifies the distinguished name to use when connecting to the LDAP server and is a null-terminated character string. An anonymous bind will be done if NULL is specified for this field.

### *LDAP\_password*

Specifies the password to use when connecting to the LDAP server and is a null-terminated character string. This field is ignored if NULL is specified for LDAP\_user.

### *LDAP\_CA\_roots*

Specifies the location of CA certificates and certificate revocation lists used to validate certificates. When GSK\_CA\_ROOTS\_LOCAL\_ONLY is specified, the CA certificates and certificate revocation lists are obtained from the local database. When GSK\_CA\_ROOTS\_LOCAL\_AND\_X500 is specified, the CA certificates and certificate revocation lists are obtained from the LDAP server if they are not found in the local database. Even when an LDAP server is used, root CA certificates must be found in the local database since the LDAP server is not a trusted data source.

### *auth\_type*

Specifies the client authentication type. This field is ignored unless LDAP\_CA\_roots is set to GSK\_CA\_ROOTS\_LOCAL\_AND\_X500. The client certificate is not validated when GSK\_CLIENT\_AUTH\_PASSTHRU is specified. The client certificate is validated using just the local database when GSK\_CLIENT\_AUTH\_LOCAL is specified. CA certificates and certificate revocation lists not found in the local database will be obtained from the LDAP server when GSK\_CLIENT\_AUTH\_STRONG or GSK\_CLIENT\_AUTH\_STRONG\_OVER\_SSL is specified (the local database must still contain the root CA certificates). There is no difference between GSK\_CLIENT\_AUTH\_STRONG and GSK\_CLIENT\_AUTH\_STRONG\_OVER\_SSL.

The **gsk\_initialize()** routine supports the following environment variables:

### **GSKV2CACHESIZE**

Specifies the number of entries in the SSL V2 session cache with a range of 0 to 32000. The value specified by the GSK\_V2\_SIDCACHE\_SIZE environment variable will be used if the GSKV2CACHESIZE variable is not defined. The default value is 256 if neither environment variable is defined.

### **GSKV3CACHESIZE**

Specifies the number of entries in the SSL V3 session cache with a range of 0 to 64000. The value specified by the GSK\_V3\_SIDCACHE\_SIZE environment variable will be used if the GSKV3CACHESIZE variable is not defined. The default value is 512 if neither environment variable is defined. The SSL V3 session cache is used for both the SSL V3 and TLS V1 protocols.

## **Related Topics**

**gsk\_secure\_soc\_init()**

**gsk\_secure\_soc\_read()**

**gsk\_secure\_soc\_write()**

**gsk\_secure\_soc\_close()**

**gsk\_uninitialize()**

## **gsk\_secure\_soc\_close**

---

### **gsk\_secure\_soc\_close**

Closes a secure socket connection.

#### **Format**

```
#include <gskssl.h>

void gsk_secure_soc_close(
    gsk_soc_data *    handle)
```

#### **Parameters**

##### **handle**

Specifies the connection handle returned by the **gsk\_secure\_soc\_init()** routine.

#### **Usage**

The **gsk\_secure\_soc\_close()** routine closes a secure connection created by the **gsk\_secure\_soc\_init()** routine. The socket itself is not closed (the application is responsible for closing the socket). The connection can no longer be used for secure communications after calling the **gsk\_secure\_soc\_close()** routine.

#### **Related Topics**

**gsk\_initialize()**

**gsk\_secure\_soc\_init()**

**gsk\_secure\_soc\_read()**

**gsk\_secure\_soc\_write()**

## gsk\_secure\_soc\_init()

Initializes a secure socket connection.

### Format

```
#include <gskssl.h>

gsk_soc_data * gsk_secure_soc_init(
    gsk_soc_init_data *    init_data)
```

### Parameters

#### init\_data

Specifies the socket connection initialization data.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### [GSK\_ERR\_INIT\_PARM\_NOT\_VALID]

A connection initialization parameter is not valid.

#### [GSK\_ERROR\_BAD\_CERT]

A certificate is not valid.

#### [GSK\_ERROR\_BAD\_DATE]

A certificate is not valid yet or is expired.

#### [GSK\_ERROR\_BAD\_MAC]

Message verification failed.

#### [GSK\_ERROR\_BAD\_MALLOC]

Insufficient storage is available.

#### [GSK\_ERROR\_BAD\_MESSAGE]

Incorrectly-formatted message received from peer application.

#### [GSK\_ERROR\_BAD\_PEER]

Peer application has violated the SSL protocol.

#### [GSK\_ERROR\_BAD\_STATE]

The SSL environment has not been initialized.

#### [GSK\_ERROR\_CRYPTO]

Cryptographic error detected.

#### [GSK\_ERROR\_GSK\_INITIALIZATION\_FAILED]

SSL connection initialization failed.

#### [GSK\_ERROR\_IO]

I/O error communicating with peer application.

#### [GSK\_ERROR\_LDAP]

An LDAP error is detected.

#### [GSK\_ERROR\_LDAP\_NOT\_AVAILABLE]

The LDAP server is not available.

#### [GSK\_ERROR\_NO\_CIPHERS]

No cipher specifications.

## **gsk\_secure\_soc\_init()**

### **[GSK\_ERROR\_SELF\_SIGNED]**

A self-signed certificate cannot be validated.

### **[GSK\_ERROR\_SOCKET\_CLOSED]**

Socket connection closed by peer application.

### **[GSK\_ERROR\_UNKNOWN\_CA]**

A certification authority certificate is missing.

### **[GSK\_ERROR\_UNSUPPORTED\_CERTIFICATE\_TYPE]**

The certificate type is not supported by System SSL.

### **[GSK\_ERROR\_VALIDATION]**

Certificate validation error.

### **[GSK\_KEYFILE\_BAD\_DNAME]**

The specified key is not found in the key database or the key is not trusted.

### **[GSK\_KEYFILE\_DUPLICATE\_NAME]**

The key database contains multiple certificates with the same subject name as the distinguished name specified in the connection initialization data.

### **[GSK\_SOC\_NO\_READ\_FUNCTION]**

No read function is specified in the connection initialization data.

### **[GSK\_SOC\_NO\_WRITE\_FUNCTION]**

No write function is specified in the connection initialization data.

## **Usage**

The **gsk\_secure\_soc\_init()** routine initializes a secure socket connection. The **gsk\_initialize()** routine must be called before any secure socket connections can be initialized. After the connection has been initialized, it can be used for secure data transmission using the **gsk\_secure\_soc\_read()** and **gsk\_secure\_soc\_write()** routines. The **gsk\_secure\_soc\_close()** routine should be called to close the connection when it is no longer needed. The **gsk\_secure\_soc\_close()** routine should not be called if an error is returned by the **gsk\_secure\_soc\_init()** routine.

Before calling the **gsk\_secure\_soc\_init()** routine, the application must create a connected socket. For a client, this means calling the **socket()** and **connect()** routines. For a server, this means calling the **socket()**, **listen()**, and **accept()** routines. However, SSL does not require the use of TCP/IP for the communications layer. The socket descriptor can be any integer value that is meaningful to the application. The application must provide its own socket routines if it is not using TCP/IP.

An SSL handshake is performed as part of the processing of the **gsk\_secure\_soc\_init()** routine. This establishes the server identity and optionally the client identity. It also negotiates the cryptographic parameters to be used for the connection.

The server certificate must support key encipherment. This means that the public/private key algorithm must use RSA encryption since the DSS (Digital Signature Standard) algorithm does not support data encryption and the certificate key usage extension (if any) must allow key encipherment.

The client certificate must support digital signatures. This means the certificate key usage extension (if any) must allow digital signature. The key algorithm can be either the RSA encryption algorithm or the Digital Signature Standard algorithm (DSA).

The SSL server always provides its certificate to the SSL client as part of the handshake. Depending upon the server handshake type, the server may ask the client to provide its certificate. The key label stored in the connection is used to retrieve the certificate from the key database or key ring. The default key will be used if no label is set. The key record must contain both an X.509 certificate and a private key.

The following SSL V2 cipher specifications are supported:

- "1" = 128-bit RC4 encryption with MD5 message authentication (128-bit secret key)
- "2" = 128-bit RC4 export encryption with MD5 message authentication (40-bit secret key)
- "3" = 128-bit RC2 encryption with MD5 message authentication (128-bit secret key)
- "4" = 128-bit RC2 export encryption with MD5 message authentication (40-bit secret key)
- "6" = 56-bit DES encryption with MD5 message authentication (56-bit secret key)
- "7" = 168-bit Triple DES encryption with MD5 message authentication (168-bit secret key)

The following SSL V3 cipher specifications are supported:

- "00" = No encryption or message authentication
- "01" = No encryption with MD5 message authentication
- "02" = No encryption with SHA-1 message authentication
- "03" = 40-bit RC4 encryption with MD5 message authentication and RSA key exchange
- "04" = 128-bit RC4 encryption with MD5 message authentication and RSA key exchange
- "05" = 128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange
- "06" = 40-bit RC2 encryption with MD5 message authentication and RSA key exchange
- "09" = 56-bit DES encryption with SHA-1 message authentication and RSA key exchange
- "0A" = 168-bit Triple DES encryption with SHA-1 message authentication and RSA key exchange
- "2F" = 128-bit AES encryption with SHA-1 message authentication and RSA key exchange
- "35" = 256-bit AES encryption with SHA-1 message authentication and RSA key exchange

The `gsk_soc_init_data` structure contains the following fields:

*fd* Specifies the socket descriptor for the secure connection. The socket must remain open until after the `gsk_secure_soc_close()` routine has been called to close the secure connection.

*hs\_type* Specifies the desired handshake type as follows:

**GSK\_AS\_CLIENT**

Performs a client SSL handshake

**GSK\_AS\_CLIENT\_NO\_AUTH**

Performs a client SSL handshake but do not provide a client certificate to the SSL server

**GSK\_AS\_SERVER**

Performs a server SSL handshake

**GSK\_AS\_SERVER\_WITH\_CLIENT\_AUTH**

Performs a server SSL handshake with client authentication

*DName* Specifies either the distinguished name or the key label of the local certificate. Specify NULL to use the default key for the key database or key ring.

*sec\_type* Returns the selected security protocol as "SSLV2", "SSLV3", or "TLSV1". This is a static string and must not be modified or freed by the application.

*cipher\_specs* Specifies the SSL V2 cipher specifications as a null-terminated string consisting of 1 or more 1-character values. Specify NULL to use the default cipher specifications ("713642" if domestic encryption is enabled and "642" otherwise). Valid cipher specifications that are not supported due to the installed cryptographic level will be skipped when the connection is initialized.

*v3cipher\_specs* Specifies the SSL V3 cipher specifications as a null-terminated string consisting of 1 or more

## **gsk\_secure\_soc\_init()**

2-character values. Specify NULL to use the default cipher specifications ("0504352F0A090306020100" if domestic encryption is enabled and "090306020100" otherwise). The SSL V3 cipher specifications are used for both the SSL V3 and TLS V1 protocols. Valid cipher specifications that are not supported due to the installed cryptographic level will be skipped when the connection is initialized.

*skread* Specifies the address of the read routine used during the SSL handshake.

*skwrite*

Specifies the address of the write routine used during the SSL handshake.

*cipherSelected*

Returns the selected cipher for the SSL V2 protocol as a 3-byte binary value:

- 0x010080 - 128-bit RC4 encryption with MD5 message authentication
- 0x020080 = 128-bit RC4 export encryption with MD5 message authentication
- 0x030080 = 128-bit RC2 encryption with MD5 message authentication
- 0x040080 = 128-bit RC2 export encryption with MD5 message authentication
- 0x060040 = 56-bit DES encryption with MD5 message authentication
- 0x0700c0 = 168-bit Triple DES encryption with MD5 message authentication

*v3cipherSelected*

Returns the selected cipher for the SSL V3 or TLS V1 protocol as a 2-byte character value with no string delimiter:

- "00" = No encryption or message authentication
- "01" = No encryption with MD5 message authentication
- "02" = No encryption with SHA-1 message authentication
- "03" = 40-bit RC4 encryption with MD5 message authentication and RSA key exchange
- "04" = 128-bit RC4 encryption with MD5 message authentication and RSA key exchange
- "05" = 128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange
- "06" = 40-bit RC2 encryption with MD5 message authentication and RSA key exchange
- "09" = 56-bit DES encryption with SHA-1 message authentication and RSA key exchange
- "0A" = 168-bit Triple DES encryption with SHA-1 message authentication and RSA key exchange
- "2F" = 128-bit AES encryption with SHA-1 message authentication and RSA key exchange
- "35" = 256-bit AES encryption with SHA-1 message authentication and RSA key exchange

*failureReasonCode*

Returns the **gsk\_secure\_soc\_init()** error code.

*cert\_info*

Returns peer certificate information. The application must not modify or free this information.

*gsk\_data*

This field is ignored. The key database information is set when **gsk\_initialize()** is called.

## **Related Topics**

**gsk\_initialize()**

**gsk\_secure\_soc\_write()**

**gsk\_secure\_soc\_read()**

**gsk\_secure\_soc\_close()**

**gsk\_get\_dn\_by\_label()**

**gsk\_secure\_soc\_init()**

| **gsk\_get\_cipher\_info()**

| **gsk\_secure\_soc\_reset()**

## `gsk_secure_soc_read()`

---

### `gsk_secure_soc_read()`

Reads data using a secure socket connection.

#### Format

```
#include <gskssl.h>

int gsk_secure_soc_read(
    gsk_soc_data *    soc_handle,
    void *            buffer,
    int               size)
```

#### Parameters

##### `soc_handle`

Specifies the connection handle returned by the `gsk_secure_soc_init()` routine.

##### `buffer`

Specifies the buffer to receive the data read from the secure socket connection. The maximum amount of data returned by `gsk_secure_soc_read()` is 16384 (16K) bytes minus the length of the SSL protocol headers.

##### `size`

Specifies the size of the supplied buffer.

#### Results

The function return value will be the number of bytes read if no error is detected. Otherwise, it will be a negative value representing one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

##### [GSK\_ERROR\_BAD\_BUFFER\_SIZE]

The buffer address or buffer size is not valid.

##### [GSK\_ERROR\_BAD\_MAC]

Message verification failed.

##### [GSK\_ERROR\_BAD\_MALLOC]

Insufficient storage is available.

##### [GSK\_ERROR\_BAD\_MESSAGE]

Incorrectly-formatted message received from peer application.

##### [GSK\_ERROR\_BAD\_PEER]

Peer application has violated the SSL protocol.

##### [GSK\_ERROR\_BAD\_SSL\_HANDLE]

The connection handle is not valid.

##### [GSK\_ERROR\_CONNECTION\_ACTIVE]

A read request is already active for the connection.

##### [GSK\_ERROR\_CRYPTO]

Cryptographic error detected.

##### [GSK\_ERROR\_IO]

I/O error communicating with peer application.

##### [GSK\_ERROR\_SOCKET\_CLOSED]

Socket connection closed by peer application.

##### [GSK\_ERROR\_WOULD\_BLOCK]

A complete SSL record is not available.

**[GSK\_ERROR\_WOULD\_BLOCK\_WRITE]**

An SSL handshake is in progress but data cannot be written to the socket.

**Usage**

The **gsk\_secure\_soc\_read()** routine reads data from a secure socket connection and returns it in the application buffer. SSL is a record-based protocol and a single call will never return more than a single SSL record (a maximum of 16384 bytes minus the length of the SSL protocol headers). The application can read an entire SSL record in a single call by supplying a buffer large enough to contain the record. Otherwise, multiple calls will be required to retrieve the entire SSL record.

SSL supports multiple threads but only one thread at a time can call the **gsk\_secure\_soc\_read()** routine for a given connection handle. Multiple concurrent threads can call **gsk\_secure\_soc\_read()** as long as each thread has its own connection handle.

SSL supports sockets in blocking mode and in non-blocking mode. When a socket is in non-blocking mode and a complete SSL record is not available, **gsk\_secure\_soc\_read()** will return with **GSK\_ERROR\_WOULD\_BLOCK**. No data will be returned in the application buffer when **GSK\_ERROR\_WOULD\_BLOCK** is returned. The application should call **gsk\_secure\_soc\_read()** again when there is data available to be read from the socket.

The peer application can initiate an SSL handshake sequence after the connection is established. If this is done and the socket is in non-blocking mode, it is possible for **gsk\_secure\_soc\_read()** to return with **GSK\_ERROR\_WOULD\_BLOCK\_WRITE**. This indicates that an SSL handshake is in progress and the application should call **gsk\_secure\_soc\_read()** again when data can be written to the socket. No data will be returned in the application buffer when **GSK\_ERROR\_WOULD\_BLOCK\_WRITE** is returned.

The application should not read data directly from the socket since this can cause SSL protocol errors if the application inadvertently reads part of an SSL record. If the application must read data from the socket, it is responsible for synchronizing this activity with the peer application so that no SSL records are sent while the application is performing its own read operations.

**Related Topics**

**gsk\_initialize()**

**gsk\_secure\_soc\_init()**

**gsk\_secure\_soc\_write()**

**gsk\_secure\_soc\_close()**

## `gsk_secure_soc_reset()`

---

### `gsk_secure_soc_reset()`

Resets the session keys for a secure connection.

#### Format

```
#include <gskssl.h>

gsk_status gsk_secure_soc_reset(
    gsk_soc_data *    soc_handle)
```

#### Parameters

##### `soc_handle`

Specifies the connection handle returned by the `gsk_secure_soc_init()` routine.

#### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the `gskssl.h` include file. The following are some common errors:

##### [GSK\_ERROR\_BAD\_MALLOC]

Insufficient storage is available.

##### [GSK\_ERROR\_BAD\_SSL\_HANDLE]

The connection handle is not valid.

##### [GSK\_ERROR\_IO]

I/O error communicating with peer application.

##### [GSK\_ERROR\_NOT\_SSLV3]

The session is not using the SSL V3 or TLS V1 protocol.

##### [GSK\_ERROR\_SOCKET\_CLOSED]

Socket connection closed by peer application.

#### Usage

The `gsk_secure_soc_reset()` routine generates new session keys for the connection. A full SSL handshake will be performed if the session has expired. Otherwise a short SSL handshake will be performed. The `gsk_secure_soc_reset()` routine can be called only for a session using the SSL V3 or TLS V1 protocol. The `gsk_secure_soc_reset()` routine initiates the SSL handshake but does not wait for it to complete. Any pending handshake messages will be processed when the `gsk_secure_soc_read()` routine is called to process incoming data.

#### Related Topics

`gsk_secure_soc_init()`

## gsk\_secure\_soc\_write()

Writes data using a secure socket connection.

### Format

```
#include <gskssl.h>

int gsk_secure_soc_write(
    gsk_soc_data *   soc_handle,
    void *           buffer,
    int              length)
```

### Parameters

**soc\_handle**

Specifies the connection handle returned by the **gsk\_secure\_soc\_init()** routine.

**buffer**

Specifies the buffer containing the data to write to the secure socket connection.

**length**

Specifies the amount to write.

### Results

The function return value will be the number of bytes written if no error is detected. Otherwise, it will be a negative value representing one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

**[GSK\_ERROR\_BAD\_BUFFER\_SIZE]**

The buffer address or buffer size is not valid.

**[GSK\_ERROR\_BAD\_MALLOC]**

Insufficient storage is available.

**[GSK\_ERROR\_BAD\_SSL\_HANDLE]**

The connection handle is not valid.

**[GSK\_ERROR\_CONNECTION\_ACTIVE]**

A write request is already active for the connection.

**[GSK\_ERROR\_CRYPTO]**

Cryptographic error detected.

**[GSK\_ERROR\_IO]**

I/O error communicating with peer application.

**[GSK\_ERROR\_SOCKET\_CLOSED]**

Socket connection closed by peer application.

**[GSK\_ERROR\_WOULD\_BLOCK]**

The SSL record cannot be written to the socket due to an EWOULDBLOCK condition.

### Usage

The **gsk\_secure\_soc\_write()** routine writes data to a secure socket connection. SSL is a record-based protocol with a maximum record length of 16384 bytes minus the length of the SSL protocol headers. Application data larger than the size of an SSL record will be sent using multiple records. Since **gsk\_secure\_soc\_read()** never returns more than a single SSL record, the receiving application will need to call **gsk\_secure\_soc\_read()** multiple times in order to receive all of the application data when multiple records are needed.

## **gsk\_secure\_soc\_write()**

SSL supports multiple threads but only one thread at a time can call the **gsk\_secure\_soc\_write()** routine for a given connection handle. Multiple concurrent threads can call **gsk\_secure\_soc\_write()** as long as each thread has its own connection handle.

SSL supports sockets in blocking mode and in non-blocking mode. When a socket is in non-blocking mode and the SSL record cannot be written to the socket, **gsk\_secure\_soc\_write()** will return with `GSK_ERROR_WOULD_BLOCK`. The application must call **gsk\_secure\_soc\_write()** again when the socket is ready to accept more data, specifying the same buffer address and buffer size as the original request. A new write request must not be initiated until the pending write request has been completed as indicated by a return value of 0.

The application should not write data directly to the socket since this can cause SSL protocol errors if the application inadvertently intermixes its data with SSL protocol data. If the application must write data to the socket, it is responsible for synchronizing this activity with the peer application so that application data is not intermixed with SSL data.

## **Related Topics**

**gsk\_initialize()**

**gsk\_secure\_soc\_init()**

**gsk\_secure\_soc\_read()**

**gsk\_secure\_soc\_close()**

## gsk\_srb\_initialize()

Initializes SRB support.

### Format

```
#include <gskssl.h>

gsk_status gsk_srb_initialize (
    int      num_tasks)
```

### Parameters

*num\_tasks*

Specifies the maximum number of service tasks and must be greater than 0.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

#### **[GSK\_ERR\_INIT\_PARM\_NOT\_VALID]**

The number of tasks parameter is not valid.

#### **[GSK\_ERROR\_BAD\_STATE]**

The SSL environment is not initialized.

#### **[GSK\_SRB\_INIT\_ESTAEX]**

Unable to establish ESTAE exit.

#### **[GSK\_SRB\_INIT\_NOT\_APF]**

The application is not APF authorized.

#### **[GSK\_SRB\_INIT\_THREAD\_CREATE]**

Unable to create a thread.

### Usage

The **gsk\_srb\_initialize()** routine will initialize the SRB (Service Request Block) support. The application must be APF-authorized in order to use SRB mode. The **gsk\_srb\_initialize()** routine must be called after the **gsk\_initialize()** routine and before any calls to the GSKSRBRD and GSKSRBWT routines.

The SRB support provided by System SSL is a mode converter which allows an SSL read or write operation to be initiated in SRB mode but processed in TASK mode. This is necessary because SRB mode is not supported by many of the functions invoked by System SSL while processing a read or write request.

The **gsk\_srb\_initialize()** routine creates a monitor thread and the first service thread. Additional threads are created as needed up to the maximum number of threads specified by the *num\_tasks* parameter. These threads will be destroyed and SRB mode support will be terminated when the **gsk\_uninitialize()** routine is called.

Refer to the z/OS Authorized Assembler Services Guide for more information about service request blocks.

### Related Topics

**GSKSRBRD**

**GSKSRBWT**

## GSKSRBRD

---

### GSKSRBRD

Reads from a secure connection in SRB mode.

#### Format

```
LOAD EP=GSKSRBRD
LR    15,0
CALL  (15), (SOCHNDLE, BUFPTR, BUFSIZE, RSNCODE)
```

#### Parameters

##### sochndle

Specifies a 4-byte word containing the `gsk_soc_data` address returned by the `gsk_secure_soc_init()` routine.

##### bufptr

Specifies a 4-byte word containing the address of the data buffer.

##### bufsize

Specifies a 4-byte word containing the length of the data buffer.

##### rsncode

Specifies a 4-byte word which will contain the reason code if an error is detected. In most cases, this will be the *errno* value at the completion of the read request.

#### Results

The return value will be the number of bytes read if no error is detected. Otherwise, it will be a negative value representing one of the return codes listed in the `gskssl.h` include file. Refer to the description of the `gsk_secure_soc_read()` routine for more information.

#### Usage

The GSKSRBRD routine is called to read from a secure connection in SRB mode. The `gsk_srb_initialize()` routine must have been called previously to initialize the SRB support. All of the parameters must be in the application storage key and must reside in the primary address space. The GSKSRBRD routine will pass the read request to one of the SRB service tasks. The service task will then call the `gsk_secure_soc_read()` routine. The GSKSRBRD routine will not return until the `gsk_secure_soc_read()` routine has completed.

#### Related Topics

[GSKSRBWT](#)

[gsk\\_initialize\(\)](#)

[gsk\\_srb\\_initialize\(\)](#)

[gsk\\_secure\\_soc\\_init\(\)](#)

[gsk\\_secure\\_soc\\_write\(\)](#)

[gsk\\_secure\\_soc\\_close\(\)](#)

---

## GSKSRBWT

Writes to a secure connection in SRB mode.

### Format

LOAD EP=GSKSRBRD

LR 15,0

CALL (15), (SOCHNDLE, BUFPTR, BUFSIZE, RSNCODE)

### Parameters

#### sochndle

Specifies a 4-byte word containing the `gsk_soc_data` address returned by the `gsk_secure_soc_init()` routine.

#### bufptr

Specifies a 4-byte word containing the address of the data buffer.

#### bufsize

Specifies a 4-byte word containing the length of the data to be written.

#### rsncode

Specifies a 4-byte word which will contain the reason code if an error is detected. In most cases, this will be the *errno* value at the completion of the read request.

### Results

The return value will be the number of bytes written if no error is detected. Otherwise, it will be a negative value representing one of the return codes listed in the `gskssl.h` include file. Refer to the description of the `gsk_secure_soc_write()` routine for more information.

### Usage

The GSKSRBWT routine is called to write to a secure connection in SRB mode. The `gsk_srb_initialize()` routine must have been called previously to initialize the SRB support. All of the parameters must be in the application storage key and must reside in the primary address space. The GSKSRBWT routine will pass the write request to one of the SRB service tasks. The service task will then call the `gsk_secure_soc_write()` routine. The GSKSRBWT routine will not return until the `gsk_secure_soc_write()` routine has completed.

### Related Topics

GSKSRBRD

`gsk_initialize()`

`gsk_srb_initialize()`

`gsk_secure_soc_init()`

`gsk_secure_soc_write()`

`gsk_secure_soc_close()`

## **gsk\_uninitialize()**

---

### **gsk\_uninitialize()**

Terminates the SSL environment.

### **Format**

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

```
gsk_status gsk_uninitialize ( void )
```

### **Parameters**

There are no parameters.

### **Results**

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following is a common error:

#### **[GSK\_ERROR\_CLOSE\_FAILED]**

An error occurred while closing the environment.

### **Usage**

The **gsk\_uninitialize()** routine will close the SSL environment created by the **gsk\_initialize()** routine. New SSL connections cannot be initiated after calling the **gsk\_uninitialize()** routine until the **gsk\_initialize()** routine is called to initialize a new SSL environment. All resources allocated for the environment will be released unless there are active SSL connections still using the environment. If there are active connections, the environment will not be actually closed until the last connection has been closed.

### **Related Topics**

**gsk\_initialize()**

**gsk\_secure\_soc\_init()**

## gsk\_user\_set()

Sets an application callback.

### Format

```
#include <gskssl.h>

gsk_status gsk_user_set(
    gsk_user_set_fid    set_id,
    void *              set_data,
    void *              reserved)
```

### Parameters

**set\_id**

Specifies the set function identifier.

**set\_data**

Specifies the address of the set data.

**reserved**

Specify NULL for this parameter.

### Results

The function return value will be 0 (**GSK\_OK**) if no error is detected. Otherwise, it will be one of the return codes listed in the **gskssl.h** include file. The following are some common errors:

**[GSK\_BAD\_PARAMETER]**

A parameter is not valid.

**[GSK\_ERROR\_BAD\_STATE]**

The SSL environment has not been initialized.

### Usage

The **gsk\_user\_set()** routine will set or reset an application callback. The **gsk\_initialize()** routine must be called before the **gsk\_user\_set()** routine can be called.

The following set function identifiers are supported:

**[GSK\_SET\_SIDCACHE\_CALLBACK]**

This function sets the session identifier cache callback. The set data is the address of the **gsk\_sidcache\_callback** structure. The application session identifier cache will be used only for SSL servers (the internal cache is always used for SSL clients). This sets the session identifier cache for existing connections as well as new connections created by the **gsk\_secure\_soc\_init()** routine.

The routine specified by the *Get* entry is called to retrieve an entry from the session identifier cache. The *session\_id* parameter is the session identifier, the *session\_id\_length* parameter is the length of the session identifier, and the *ssl\_version* parameter is the SSL protocol version number (**GSK\_SSLVERSION\_V2** or **GSK\_SSLVERSION\_V3**). The function return value is the address of the session data buffer or NULL if an error is detected. The *FreeDataBuffer* routine will be called to release the session data buffer when it is no longer needed by the SSL runtime.

```
gsk_data_buffer * Get (
    const unsigned char *    session_id,
    unsigned int            session_id_length,
    gsk_sslversion          ssl_version)
```

## **gsk\_user\_set()**

The routine specified by the *Put* entry is called to store an entry in the session identifier cache. The *ssl\_session\_data* parameter is the session data, the *session\_id* parameter is the session identifier, the *session\_id\_length* parameter is the length of the session identifier, and the *ssl\_version* parameter is the SSL protocol version number (GSK\_SSLVERSION\_V2 or GSK\_SSLVERSION\_V3). The function return value is ignored and can be a NULL address. The callback routine must make its own copy of the session data since the SSL structure will be released when the connection is closed.

```
gsk_data_buffer * Put (
    gsk_data_buffer *    ssl_session_data,
    const unsigned char * session_id,
    unsigned int         session_id_length,
    gsk_sslversion      ssl_version)
```

The routine specified by the *Delete* entry is called to remove an entry from the session identifier cache. The *session\_id* parameter is the session identifier, the *session\_id\_length* parameter is the length of the session identifier, and the *ssl\_version* parameter is the SSL protocol version number (GSK\_SSLVERSION\_V2 or GSK\_SSLVERSION\_V3).

```
void Delete (
    const unsigned char * session_id,
    unsigned int         session_id_length,
    gsk_sslversion      ssl_version)
```

The routine specified by the *FreeDataBuffer* entry is called to release the data buffer returned by the *Get* routine.

```
void FreeDataBuffer (
    gsk_data_buffer *    ssl_session_data)
```

## **[GSK\_RESET\_SIDCACHE\_CALLBACK]**

This function resets the session identifier cache callback. The internal session identifier cache will be used instead of an application session identifier cache. This resets the session identifier cache for existing connections as well as new connections created by the **gsk\_secure\_soc\_init()** routine.

## **[GSK\_SET\_GETPEER\_CALLBACK]**

This function sets the peer identification callback. The peer identification callback returns the 32-bit network identifier for the remote partner. The *fd* parameter is the socket descriptor specified when the connection was initialized. The peer identification routine will be called for new connections created by **gsk\_secure\_soc\_init()** but will not be called for existing connections.

```
unsigned long io_getpeerid (
    int    fd)
```

## **[GSK\_RESET\_GETPEER\_CALLBACK]**

This function resets the peer identification callback. The internal peer identification routine will be used instead of the application routine. This applies to new connections created by **gsk\_secure\_soc\_init()** and does not affect existing connections.

---

## Chapter 10. Certificate/Key Management

This chapter discusses the use of the z/OS shell-based **gskkyman** command to manage private keys and certificates. In addition, detailed examples using the **gskkyman** command are shown in this chapter (see “Example Tasks Performed by the gskkyman Command in Interactive Mode” on page 259).

---

### Introduction

SSL connections make use of public/private key mechanisms for authenticating each side of the SSL session and agreeing on bulk encryption keys to be used for the SSL session. To use public/private key mechanisms (termed PKI), public/private key pairs must be generated. In addition, X.509 certificates (which contain public keys) may need to be created, or certificates must be requested, received, and managed.

System SSL supports the following two methods for managing PKI private keys and certificates:

- A z/OS shell-based program called **gskkyman**. **gskkyman** creates, fills in, and manages a z/OS HFS file that contains PKI private keys, certificate requests, and certificates. This z/OS HFS file is called a **key database** and, by convention, has a file extension of **.kdb**.
- The z/OS Security Server (RACF) RACDCERT command. RACDCERT installs and maintains PKI private keys and certificates in RACF. Refer to the *z/OS: Security Server RACF Command Language Reference*, SA22-7687 for details on the RACDCERT command. RACF supports multiple PKI private keys and certificates to be managed as a group. These groups are called **key rings**. RACF key rings are the preferred method for managing PKI private keys and certificates for System SSL.

The System SSL application uses the GSK\_KEYRING\_FILE parameter of the **gsk\_attribute\_set\_buffer()** API or the GSK\_KEYRING\_FILE environment variable to specify the locations of the PKI private keys and certificates to System SSL. If you are using a z/OS HFS key database, the key database file name is passed in this parameter. If you are using a RACF key ring, the name of the key ring is passed in this parameter.

---

### The gskkyman Command

**gskkyman** is a z/OS shell-based program that creates, fills in, and manages a z/OS HFS file that contains PKI private keys, certificate requests, and certificates. This z/OS HFS file is called a **key database** and, by convention, has a file extension of **.kdb**.

The interface to **gskkyman**, while command-line based, is an interactive dialog between you (the user) and the program. At each step, the **gskkyman** program prompts you with one or more lines of output and expects a numeric choice to be supplied as input at the prompt. Once a choice has been made, the **gskkyman** program prompts you for the individual pieces of information needed to fulfill the request. You are prompted for each piece of information. Many times there is a default choice that is listed between parentheses at the end of the command prompt. If the default choice is acceptable, press Enter to select the default. If a choice other than the default is desired, enter the value at the prompt and press Enter. If a value is entered that is outside of the acceptable range of inputs, you will be re-prompted for the information.

---

## Setting Up the Environment to Run gskkyman

**gskkyman** uses the DLLs that are installed with System SSL and must have access to these at run-time. **gskkyman** must also have access to the message catalogs. The /bin directory includes a symbolic link to **gskkyman**, therefore, if your PATH environment variable contains this directory, you will find **gskkyman**. If your PATH environment variable does not contain this directory, add /usr/lpp/gskssl/bin to your PATH using the following:

```
PATH=$PATH:/usr/lpp/gskssl/bin
```

/usr/lib/nls/msg/En\_US.IBM-1047 (as well as /usr/lib/nls/msg/Ja\_JP.IBM-939 for JCPT34J installations) include symbolic links to the message catalogs for **gskkyman**. If they do not include these links, add /usr/lpp/gskssl/lib/nls/msg to your NLSPATH using the following command:

```
export NLSPATH=$NLSPATH:/usr/lpp/gskssl/lib/nls/msg/%L/%N
```

This setting assumes that your environment has the LANG environment variable set to En\_US.IBM-1047 (or Ja\_JP.IBM-939 for JCPT34J installations that expect Japanese messages and prompts). If LANG is not set properly, set the NLSPATH environment variable using the following command:

```
export NLSPATH=/usr/lpp/gskssl/lib/nls/msg/En_US.IBM-1047/%N:$NLSPATH
```

or for JCPT34J installations that expect Japanese messages and prompts:

```
export NLSPATH=/usr/lpp/gskssl/lib/nls/msg/Ja_JP.IBM-939/%N:$NLSPATH
```

The DLLs for System SSL are installed into a partitioned dataset (PDS). These DLLs are **not** installed into the LINKLIB or LPALIB by default. To access these DLLs, if they have not been placed in LINKLIB or LPALIB, you must set the STEPLIB environment variable to find the DLLs. Consult your system programmer for the high-level qualifier of the System SSL PDS. In this example, the high-level qualifier for the System SSL PDS is <GSKHLQ>. In the following command, replace <GSKHLQ> with the value for your installation:

```
export STEPLIB=$STEPLIB:<GSKHLQ>.SGSKLOAD
```

---

## Key Database Files

Key database files are password protected because they contain the private keys that are associated with some of the certificates that are contained in the key database. Private keys, as their name implies, should be protected because their value is used in verifying the authenticity of requests made during PKI operations.

It is recommended that key database files be set with the following string z/OS HFS file permissions:

```
-rw----- (600) (read-write for only the owner of the key database)
```

The owner of the key database should be the user who will be managing the key database. The program using System SSL (and the key database) must have at least read permission to the key database file at run-time. If the program is a server program that runs under a different user ID than the administrator of the key database file, it is recommended that a group be setup to control access to the key database file. In this case, it is recommended that you set the z/OS HFS permissions on the key database file to the following:

```
-rw-r---- (640) (read-write for owner and read-only for group)
```

The owner of the z/OS HFS key database file is set to the administrator user ID and the group owner of the key database file is set to the group that contains the server that will be using the key database file.

---

## How gskkyman Works

This section describes gskkyman, and how to use it.

## gskkyman

The gskkyman command is used for key database management.

### Format

**gskkyman**

**gskkyman -e -k filename -l label -p filename**

**gskkyman -g -x days -cr filename -ct filename -k filename -l label -ca**

**gskkyman -h**

**gskkyman -i -k filename -l label -p filename**

**gskkyman -s -k filename**

### Functions

**-e** Export a certificate and its associated private key

**-g** Sign a certificate for a certificate request

**-h** Display the command syntax

**-i** Import a certificate and its associated private key

**-s** Store the database password in the stash file

### Options

**-ca**

A certification authority certificate will be generated if **-ca** is specified. An end user certificate will be generated if **-ca** is not specified.

**-cr**

Specifies the name of the certificate request file. You will be prompted for the file name if this option is not specified.

**-ct** Specifies the name of the signed certificate file. You will be prompted for the file name if this option is not specified.

**-k** Specifies the name of the key database. You will be prompted for the file name if this option is not specified. The length of the fully-qualified file name cannot exceed 251 characters. If the file name does not end with an extension of 1-3 characters, the length of the fully-qualified file name cannot exceed 247 characters. Finally, the key database name cannot end with `.rdb` or `.sth`.

**-l** Specifies the record label. The label must be enclosed in double quotes if it contains one or more spaces. The label for the default key will be used if this option is not specified (export or sign function) or you will be prompted for the label (import function).

**-p** Specifies the name of the PKCS #12 file. You will be prompted for the file name if this option is not specified.

**-x** Specifies the number of days until the signed certificate expires and must be between 1 and 9999 days. The certificate will expire in 365 days if this option is not specified.

### Usage

The gskkyman command is used to manage a key database and its associated request database. Interactive menus will be displayed if no command options are specified. Otherwise, the requested database function will be performed and the gskkyman command will exit.

The key database contains certificates and private keys and normally has a filename extension of `'.kdb'`. The request database contains requests for new certificates and always has a filename extension of `'.rdb'`. The database stash file contains the masked database password and always has a filename extension of `'.sth'`. Access to these files should be restricted to the database owner.

| A certificate or request database consists of fixed-length records. The record length is specified when the database is created and must be large enough to contain the largest certificate entry. A record length of 2500 should be sufficient for most applications. The record length can be increased if necessary after the database has been created.

| A temporary database file is created when a database is updated during gskkyman processing. The temporary database file is created using the same name as the database file with ".new" appended to the name. The database file is then rewritten and the temporary database file is deleted upon successful completion of the rewrite operation. The temporary database file will not be deleted if an error occurs while rewriting the database file. If this happens, you can replace the database file with the temporary database file in order to recover from the error. If an error does occur and you do not rename or delete the temporary file, you will get an error on the next database update operation indicating the backup file already exists.

---

## Interactive Mode

Interactive mode is entered when the gskkyman command is entered without any parameters. A series of menus will be presented to allow you to select the database functions to be performed. Leading and trailing blanks will be removed from data entries but imbedded blanks will be retained. Blanks will not be removed from passwords.

## Database Menu

This is the top-level menu and is displayed when the gskkyman command starts:

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database

0 - Exit program

Enter option number:
====>
```

Figure 2. Database Menu

### Create new database

This option will create a new key database and the associated request database. You will be prompted to enter the key database name, the database password, the password expiration interval, and the database record length.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when creating the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

The database password must be between 1 and 128 characters. A password exceeding 128 characters will be truncated to 128 characters.

The password expiration interval must be between 0 and 9999 days (a value of 0 indicates the password does not expire).

The record length must be large enough to contain the largest certificate to be stored in the database and must be between 2500 and 65536.

Two files will be created: the key database and the request database with an extension of '.rdb'. The file access permissions will be set so only the owner has access to the files.

### Open database

This option will open an existing database. You will be prompted to enter the key database name and the database password.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

### **Change database password**

This option will change the database password. You can change the password at any time but you must change it once it has expired in order to access the database once more. You will be prompted to enter the key database name, the current database password, the new database password, and the new password expiration interval.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

The new database password must be between 1 and 128 characters.

The password expiration interval must be between 0 and 9999 days (a value of 0 indicates the password does not expire).

### **Change database record length**

This option will change the database record length. All database records have the same length and database entries cannot span records. You can increase the record length if you find it is too small to store a new certificate. You can decrease the record length to reduce the database size if the original record length is too large. You cannot reduce the record length to a value smaller than the largest certificate currently in the database. You will be prompted to enter the key database name, the database password, and the new record length.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

### **Delete database**

This option will delete the key database, the associated request database, and the database password stash file. You will be prompted to enter the key database name.

The fully-qualified key database name must be between 2 and 251 characters and should either have no extension or an extension of '.kdb' (the maximum database name is 247 characters if the name does not end with an extension of 1-3 characters to allow for the addition of an extension when accessing the request database or the password stash file). The key database name may not end with '.rdb' or '.sth' as these extensions are reserved for the request database and the password stash file.

## **Key Management Menu**

The key management menu is displayed once the key database has been created or opened. The key database and the associated request database are opened for update and remain open until you return to the main menu.

```
Key Management Menu

Database: Database_name

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
===>
```

Figure 3. Key Management Menu

### Manage Keys and Certificates

This option manages certificates with private keys. A list of key labels is displayed. Pressing the ENTER key without making a selection will display the next set of labels. Selecting one of the label numbers will display the following menu:

```
Key and Certificate Menu

Label: Certificate_label_name

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label

0 - Exit program

Enter option number (press ENTER to return to previous menu):
===>
```

Figure 4. Key and Certificate Menu

#### Show certificate information

This option displays information about the X.509 certificate associated with the private key.

#### Show key information

This option displays information about the private key.

#### Set key as default

This option makes the current key the default key for the database.

### Set certificate trust status

This option sets or resets the trusted status for the X.509 certificate. A certificate cannot be used for authentication unless it is trusted.

### Copy certificate and key to another database

This option copies the certificate and key to another database. An error will be returned if the certificate is already in the database or if the label is not unique.

### Export certificate to a file

This option exports just the X.509 certificate to a file. The supported export formats are ASN.1 Distinguished Encoding Rules (DER) and PKCS #7 (Cryptographic Message Syntax)

### Export certificate and key to a file

This option exports the X.509 certificate and its private key to a file. The private key is encrypted when it is written to the file. The password you select will be needed when you import the file. The supported export formats are PKCS #12 Version 1 and PKCS #12 Version 3. The PKCS #12 Version 1 format is obsolete but is the only format supported by some SSL implementations (For example, z/OS System SSL Version 1 Release 2 and earlier). The strong encryption option uses Triple DES to encrypt the private key while the export encryption option uses 40-bit RC2. The export file will contain the requested certificate and its certification chain.

### Delete certificate and key

The certificate and its associated private key are deleted.

### Change label

This option will change the label for the database record.

## Manage Certificates

This option manages certificates without private keys. A list of key labels is displayed. Pressing the ENTER key without making a selection will display the next set of labels. Selecting one of the label numbers will display the following menu:

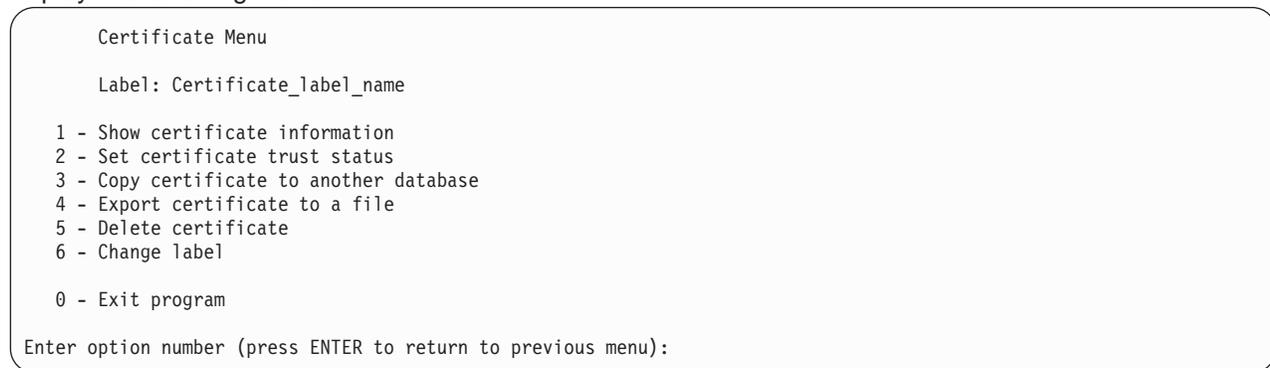


Figure 5. Certificate Menu

### Show certificate information

This option displays information about the X.509 certificate.

### Set certificate trust status

This option sets or resets the trusted status for the X.509 certificate. A certificate cannot be used for authentication unless it is trusted.

### Copy certificate to another database

This option copies the certificate to another database. An error will be returned if the certificate is already in the database or if the label is not unique.

### **Export certificate to a file**

This option exports the X.509 certificate to a file. The supported export formats are ASN.1 Distinguished Encoding Rules (DER) and PKCS #7 (Cryptographic Message Syntax)

### **Delete certificate**

The certificate is deleted.

### **Change label**

This option will change the label for the database record.

## **Manage Certificate Requests**

This option manages certificate requests. A list of request labels is displayed. Pressing the ENTER key without making a selection will display the next set of labels. Selecting one of the label numbers will display the following menu:

```
Request Menu

Label: label_name

1 - Show key information
2 - Export certificate request to a file
3 - Delete certificate request and key
4 - Change label

0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

*Figure 6. Request Menu*

### **Show key information**

This option displays information about the private key associated with the certificate request.

### **Export certificate request to a file**

This option exports the certificate request to a file in Base64 format. This file can then be sent to a certification authority for processing.

### **Delete certificate request and key**

The certificate request and its associated private key are deleted.

### **Change label**

This option will change the label for the database record.

## **Create New Certificate Request**

This option will create a certificate request using either RSA encryption or DSA for the public and private keys. The certificate request will be exported to a file in Base64 format. This file can then be sent to a certification authority for processing.

The label has a maximum length of 127 characters and is used to reference the certificate in the request database. The label will also be used when the certificate is received, so it must be unique in both the request and key databases. It must consist of characters which can be represented as 7-bit ASCII characters (letters, numbers, and punctuation) in the ISO8859-1 code page.

## **Receive Certificate Issued for your Request**

This option will receive the signed certificate returned by the certification authority. This certificate can be either a new certificate issued in response to a certificate request or a renewal of an existing certificate. If this is a new certificate, the certificate request must still be in the request database. If this is a renewal certificate, the old certificate must still be in the key database and must have the same issuer name, subject name, and public key.

The certificate file must contain either an ASN.1 DER-encoded sequence as defined in RFC 2459 (X.509 Public Key Infrastructure) or a signed data message as defined in PKCS #7 (Cryptographic Message Syntax). The data can either be the binary value or the Base64 encoding of the binary value. If the import file is in PKCS #7 format, only the first certificate will be imported. If the first certificate is not the request certificate, the import will fail with 'unable to locate matching request'.

Base64 data is in the local code page. A DER-encoded sequence must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'. A PKCS #7 signed data message must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----' or start with the encoding header '-----BEGIN PKCS #7 SIGNED DATA-----' and end with the encoding footer '-----END PKCS #7 SIGNED DATA-----'.

The request database entry will be deleted once the certificate has been received.

### **Create a Self-Signed Certificate**

This option will create a self-signed certificate using either RSA encryption or DSA for the public and private keys. The certificate can be created for use by a certification authority or an end user. A CA certificate can be used to sign other certificates and certificate revocation lists while an end user certificate can be used for authentication, digital signatures, and data encryption.

The label has a maximum length of 127 characters and is used to reference the certificate in the key database. It must consist of characters which can be represented as 7-bit ASCII characters (letters, numbers, and punctuation) in the ISO8859-1 code page.

The number of days until the certificate expires must be between 1 and 9999.

### **Import a Certificate**

This option imports a certificate and adds it to the key database. The import file contains a certificate without a private key. The certificate will be marked as trusted when it is added to the database.

The import file must contain either an ASN.1 DER-encoded sequence as defined in RFC 2459 (X.509 Public Key Infrastructure) or a signed data message as defined in PKCS #7 (Cryptographic Message Syntax). The data can either be the binary value or the Base64 encoding of the binary value. If the import file is in PKCS #7 format, only the first certificate will be imported.

Base64 data is in the local code page. A DER-encoded sequence must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'. A PKCS #7 signed data message must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----' or start with the encoding header '-----BEGIN PKCS #7 SIGNED DATA-----' and end with the encoding footer '-----END PKCS #7 SIGNED DATA-----'.

A root certificate is a self-signed certificate and will be imported as long as the certificate is not already in the key database.

An intermediate certificate is a certificate signed by another entity. The key database must already contain a certificate for the issuer. The certificate will not be imported if the certificate authenticity cannot be validated or if the database already contains the certificate.

An existing certificate can be replaced by specifying the label of the existing certificate. The issuer name, subject name, and subject public key in the new certificate must be the same as the existing certificate. If the existing certificate has a private key, the private key is not changed when the certificate is replaced.

### **Import a Certificate and a Private Key**

This option imports a certificate and the associated private key and adds it to the key database. The certificate will be marked as trusted when it is added to the database.

The import file must contain an ASN.1 DER-encoded sequence as defined in PKCS #12 (Personal Information Exchange Syntax). The data can be either the binary value or the Base64 encoding of the binary value. Base64 data is in the local code page and must start with the encoding header '-----BEGIN CERTIFICATE-----' and end with the encoding footer '-----END CERTIFICATE-----'.

A root certificate is a self-signed certificate and will be imported as long as the certificate is not already in the key database.

An intermediate CA or end entity certificate is a certificate signed by another entity. The key database must already contain a certificate for the issuer. The certificate will not be imported if the certificate authenticity cannot be validated or if the database already contains the certificate.

Each certificate in the certification chain will be imported if it is present in the import file. The certificate subject name will be used as the label for certificates added from the certification chain. A certification chain certificate will not be added to the database if the label is not unique or if the certificate is already in the database.

### **Show the Default Key**

The private key information for the default key is displayed.

### **Store Database Password**

The database password is masked and written to the key stash file. The file name is the same as the key database file name but has an extension of '.sth'.

### **Show Database Record Length**

The database record length is displayed. All records in the database have the same length and a database entry cannot span a database record.

---

## **Example Tasks Performed by the gskkyman Command in Interactive Mode**

**gskkyman** can be run from either an rlogin z/OS shell environment or from the OMVS shell command-line environment. The examples that follow were performed from the rlogin environment. If you use the OMVS shell command-line environment, the only difference is that all input will be done at the command prompt at the bottom of the screen.

The following tasks will be performed in this section:

- Creating, opening and deleting a key database file
- Changing a key database password
- Storing an encrypted key database password
- Creating a self-signed server or client certificate
- Creating a certificate request and processing the signed request
- Managing keys and certificates:
  - Show certificate/key information
  - Marking a certificate (and private key) as the default certificate for the key database
  - Copying a certificate (and private key) to a different key database:
    - Copying a certificate without its private key
    - Copying a certificate with its private key
    - Copying a certificate with its private key to a key database on the same system
  - Removing a certificate (and private key) from a key database
  - Changing a certificate label
- Importing a certificate from a file as a trusted CA certificate

- Importing a certificate from a file with its private key
- Using gskkyman to be your own certificate authority (CA)
- Migrating key database files to RACF key rings

## Starting gskkyman

To start **gskkyman**, enter **gskkyman** at the command prompt (see Figure 7).

**Note:** In the examples that follow, your input is shown in **bold**, and places where you press the Enter key are noted with **<enter>**.

Figure 7 shows the gskkyman start menu.

```
# gskkyman <enter>

      Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database

0 - Exit program

Enter option number:
===>
```

Figure 7. Starting Menu for gskkyman

From the Database Menu for **gskkyman**, you can create a new key database, open an existing key database, change a database password, change a database record length, delete a database, or exit **gskkyman**.

## Creating, Opening and Deleting a Key Database File

To create a new key database, enter 1 at the command prompt on the Database Menu:

```
      Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database

0 - Exit program

Enter option number: 1 <enter>
Enter key database name (press ENTER to return to menu): mykey.kdb <enter>
Enter database password (press ENTER to return to menu): <enter password>
Re-enter database password: <enter password>
Enter password expiration in days (press ENTER for no expiration): 35 <enter>
Enter database record length (press ENTER to use 2500): <enter>

Key database /home/sufw11/ssl_cmd/mykey.kdb created.

Press ENTER to continue.
===>
```

Figure 8. Creating a New Key Database

Figure 8 on page 260 shows the input prompts that **gskkyman** produces when you choose 1 to create a new key database. As you can see, default choices are listed in parentheses. In the example, by pressing the Enter key at the Enter database record length prompt, the default of 2500 was chosen.

**Note:** The maximum length of the password specified for a key database file is 128 characters.

After entering the database record length, a message displays confirming that your database was created (see Figure 8 on page 260). You are prompted to press Enter to continue. Doing so displays the Key Management Menu for the database you have created:

```
Key Management Menu

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
===>
```

Figure 9. Key Management Menu for **gskkyman**

Figure 9 shows the Key Management Menu. Entering 0 at this prompt exits the **gskkyman** program. Pressing Enter at the prompt returns you to the Database Menu.

To open an existing key database file, on the Database Menu, enter option number 2 (see Figure 10 on page 262). You are then prompted for the key database name and password.

**Note:** Do not lose the key database password. There is no method to reset this password if you lose or forget the password. If the password is lost, the private keys stored in the key database are inaccessible, therefore, unusable.

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database

0 - Exit program

Enter option number: 2 <enter>
Enter key database name (press ENTER to return to menu): mykey.kdb <enter>
Enter database password (press ENTER to return to menu): <enter password>

===>
```

Figure 10. Opening an Existing Key Database File

The key database name is the z/OS HFS file name of the key database. The input file name is interpreted relative to the current directory when **gskkyman** is invoked. You may also specify a fully qualified key database name .

After you enter the key database name and password, the Key Management Menu displays for the database you have selected to open, (see Figure 11).

```
Key Management Menu

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
===>
```

Figure 11. Key Management Menu

To delete an existing database, from the Database Menu, select option 5 (see Figure 12):

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database

0 - Exit program

Enter option number: 5 <enter>
Enter key database name (press ENTER to return to menu): mykey.kdb <enter>

Enter 1 to confirm delete, 0 to cancel delete: 1 <enter>

Key database /home/sufw11/ssl_cmd/mykey.kdb deleted.

Press ENTER to continue.
===>
```

Figure 12. Deleting an Existing Key Database

You are prompted to enter the key database name that you wish to delete. Then you must enter 1 to confirm the delete, or 0 to cancel the delete. If you choose 1, a message displays to confirm the file has been deleted.

**Note:** If you delete an existing key database, the associated request database and database password stash file (if existent) **will also be deleted**. It's important to note that anyone with write access to a key database can delete that database either by removing it with the **rm** command or by using **gskkyman** subcommand.

## Changing a Key Database Password

You can change a key database password. From the Database Menu, select option 3:

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database

0 - Exit program

Enter option number: 3 <enter>
Enter key database name (press ENTER to return to menu): mykey.kdb <enter>
Enter database password (press ENTER to return to menu): <enter current password>
Enter new database password (press ENTER to return to menu): <enter new password>
Re-enter database password: <enter new password>
Enter password expiration in days (press ENTER for no expiration): <enter>

Database password changed.

Press ENTER to continue.
===>
```

Figure 13. Changing a Key Database Password

Figure 13 displays the prompts you are given. You first enter your current password. Then you select a new password, and enter it again to confirm. You can choose your password expiration in days or press Enter to have no expiration. A message displays to confirm the transaction.

## Storing an Encrypted Key Database Password

In order for applications to use the key database file, the application must specify both the file name as well as its associated password. The password can either be specified directly or through a stash file containing the encrypted password. The stash file provides a level of security where the password does not have to be explicitly specified. To save the encrypted key database password, enter option 10 from the Key Management Menu:

**Note:** In the following task descriptions, it is assumed that you have opened the key database and are displaying the Key Management Menu panel.

#### Key Management Menu

Database: /home/sufw11/ssl\_cmd/mykey.kdb

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive certificate issued for your request
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password
- 11 - Show database record length
  
- 0 - Exit program

Enter option number (press ENTER to return to previous menu): 10 <enter>

Database password stored in /home/sufw11/ssl\_cmd/mykey.sth.

Press ENTER to continue.

====>

Figure 14. Storing a Database Password in a Stash File

Figure 14 shows the message you receive after entering option 10 to store the database password. In this example, the database password was stored in a file called `mykey.sth`.

## Creating a Self-Signed Server or Client Certificate

If your organization does not use a certificate authority (within the organization or outside the organization), a self-signed certificate can be generated for use by the program acting as an SSL server or client. Programs acting as SSL servers (i.e. acting as the server side of the SSL handshake protocol) must have a certificate to use during the handshake protocol. A program acting as an SSL client requires a certificate when the SSL server requests client authentication as part of the SSL handshake.

**Note:** This is not recommended for production environments and should only be used to facilitate test environments prior to production. Self-signed certificates do not imply any level of security or authenticity of the certificate because, as their name implies, they are signed by the same key that is contained in the certificate. On the other hand, certificates that are signed by a certificate authority indicate that, at least at the time of signature, the certificate authority approved the information contained in the certificate.

**Note:** `gskkyman` supports the creation of X.509 Version 3 certificates.

When creating a self-signed certificate to be used to identify a server or client, from the Key Management Menu, enter 6. You will be prompted for a number of items to define the certificate. First you will be asked to select the type of certificate to be created.

```
Key Management Menu

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 6 <enter>
===>
```

Figure 15. Select 6 to Create a Self-Signed Certificate

Certificates that are intended to be used directly by a server or client are considered to be end-user certificates. Certificates intended to be used to sign other certificates are considered to be CA certificates. RSA key certificates are the most common. DSA key certificates represent certificates that follow the FIPS-186 government standard. The larger the key size, the more secure the generated key will be. The most commonly used size is 1024. Note that CPU usage increases as the key size increases. For example, CPU usage will increase by a factor of 6 when the key size is doubled.

Once the certificate type is determined, you will be prompted to enter:

- a label to uniquely identify the key and certificate within the key database
- the individual fields within the subject name
- certificate expiration. The valid expiration range for a self-signed certificate is 1 to 9999 days. The default value is 365 days.

Figure 16 on page 266 shows the creation of a server self-signed certificate.

#### Certificate Type

- 1 - CA certificate with 1024-bit RSA key
- 2 - CA certificate with 2048-bit RSA key
- 3 - CA certificate with 1024-bit DSA key
- 4 - End user certificate with 1024-bit RSA key
- 5 - End user certificate with 2048-bit RSA key
- 6 - End user certificate with 1024-bit DSA key

```
Select certificate type (press ENTER to return to menu): 4 <enter>
Enter label (press ENTER to return to menu): Server Cert <enter>
Enter subject name for certificate
  Common name (required): My Server Certificate <enter>
  Organizational unit (optional): ID <enter>
  Organization (required): IBM <enter>
  City/Locality (optional): Endicott <enter>
  State/Province (optional): NY <enter>
  Country/Region (2 characters - required): US <enter>
Enter number of days certificate will be valid (default 365): 244 <enter>

Please wait .....

Certificate created.

Press ENTER to continue.
===>
```

Figure 16. Creating a Self-Signed Certificate

Once the certificate is created, the next step is to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by the SSL APIs without having to specify its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 272.

In order for the SSL handshake to successfully validate the use of the self-signed certificates, the partner application needs to know about the signer of the certificate. For self-signed certificates, this means the self-signed certificate must be imported into the partner's database. For more information on importing certificates, see "Importing a Certificate from a File as a Trusted CA Certificate" on page 277.

## Creating a Certificate Request and Processing the Signed Request

A program may require a certificate, associated with itself, depending on what side of the SSL connection the program is running. This requirement also depends on whether *client authentication* is requested as part of the SSL handshake. Programs acting as SSL servers (act as the server side of the SSL handshake protocol) **must** have a certificate to use during the handshake protocol. A program acting as an SSL client requires a certificate in the key database if the SSL server requests *client authentication* as part of the SSL handshake operation. The way in which certificates are used within an organization will determine whether you need to create a certificate request. If the organization chooses to use a certificate authority (within the organization or outside of the organization), then you must generate a certificate request.

To create a certificate request, enter 4 from the Key Management Menu.

```

Key Management Menu

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 4 <enter>
===>

```

Figure 17. Select 4 to Create a New Certificate Request

The Certificate Type menu appears:

```

Certificate Type

1 - Certificate with 1024-bit RSA key
2 - Certificate with 2048-bit RSA key
3 - Certificate with 1024-bit DSA key

Enter certificate type (press ENTER to return to menu): 1 <enter>
Enter request file name (press ENTER to return to menu): certreq.arm <enter>
Enter label (press ENTER to return to menu): Test Server Cert <enter>
Enter subject name for certificate
Common name (required): Test Server <enter>
Organizational unit (optional): ID <enter>
Organization (required): IBM <enter>
City/Locality (optional): Endicott <enter>
State/Province (optional): NY <enter>
Country/Region (2 characters - required): US <enter>

Please wait .....

Certificate request created.

Press ENTER to continue.
===>

```

Figure 18. Creating a Certificate Request

When creating a certificate request, you are first prompted for the type of certificate to be requested. RSA key certificates are the most common. DSA key certificates represent certificates that follow the FIPS-186 government standard. The larger the key size, the more secure the encryption/decryption generated key will be. The most commonly used size is 1024.

After the certificate type is determined, you will be prompted to enter:

- a request file name to store the certificate request
- a label to uniquely identify the certificate request within the key database
- the individual fields within the subject name.

Once the certificate request is created, a file with the name you specified will exist in the current working directory or directory specified in the filename. If you choose to exit **gskkyman**, the program ends. Otherwise, the Key Management Menu (see Figure 11 on page 262) displays, allowing additional operations to be performed.

The certificate request created is stored in a file that is in base64-encoded format. This format is what is typically required by certificate authorities that create certificates. The following is the contents of the file created by the steps performed in Figure 18 on page 267:

```
$ cat certreq.arm<enter>
-----BEGIN NEW CERTIFICATE REQUEST-----
MIH7MIgMgAgEAMEExCzAJBgNVBAYTA1VTMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsT
CEVuZG1jb3R0MREwDwYDVQQDEwhKb2huIERvZTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCrIZdRnXhH1EMAwTuKMKYznCFp4CFk0YG66BhvMGgfTwq19aSRWkVcer8I
I7Qk9aYzQ2LIpRh1oJ9ugo.jy1I9VAgMBAAGgADANBgkqhkiG9w0BAQQFAANBAFc1
x0funjyt54dUqGDdPgbnMr5A3trUhzHHkX8x1fH9A1brpsv2a3FjvnmYWFPUFXAf
3ABCD5nnsbk3AP++ic5UTM=
-----END NEW CERTIFICATE REQUEST-----
$
```

Figure 19. Contents of certreq.arm after Certificate Request Generation

This file can either be transferred to another system (as a text file) and then transferred to the certificate authority or placed directly into a mail message sent to a certificate authority using cut-and-paste methods.

In addition to the certificate request file that is generated, a request database (.rdb) file is also created or altered. The request database file will be named the same as the key database file, except it will have an extension of .rdb. For example, a key database file of key.kdb will cause a request database file of key.rdb to be created. This request database file must be saved along with the key database in order for the response for the certificate request to be successfully processed.

Once a certificate is created by the certificate authority in response to the request, you must receive it into the key database.

To receive the certificate, you must store the Base64-encoded certificate in an HFS file on the z/OS system to be read in by the **gskkyman** command. This file should be in the current working directory when **gskkyman** is started. If this file is on another working directory you will have to specify the fully qualified name.

**Note:** In order to receive the certificate the CA certificate must also exist in the key database. To store a CA certificate refer to “Importing a Certificate from a File as a Trusted CA Certificate” on page 277.

To receive a certificate issued on your behalf, from the Key Management Menu (see Figure 11 on page 262), enter 5.

```
Key Management Menu

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 5 <enter>

Enter certificate file name (press ENTER to return to menu): signed.arm <enter>

Certificate received.

Press ENTER to continue.
===>
```

Figure 20. Receiving a Certificate Issued for your Request

You are prompted for the name of the file that contains the Base64-encoded certificate that was returned to you by the certificate authority in response to a previously submitted certificate request (See “Creating a Certificate Request and Processing the Signed Request” on page 266). After receiving the certificate, you press Enter to continue working with the Key Management Menu. Upon completion of this step and prior to the System SSL APIs using the certificate during the SSL handshake processing, you need to determine whether the certificate should be marked as the database’s default certificate. Setting the certificate as the default certificate allows the certificate to be used by the SSL APIs without having to specify its label. For more information on setting the default certificate, see “Marking a Certificate (and Private Key) as the Default Certificate for the Key Database” on page 272.

## Managing Keys and Certificates

Once certificates are added to the key database, the following are some common operations that can be performed with the certificates.

- Show certificate/key information
- Mark a certificate (and private key) as the default certificate for the key database
- Export a certificate to a file or key database
- Remove a certificate (and private key ) from a key database
- Change a certificate label

### Showing Certificate/Key Information

It is sometimes useful to display the information contained in the certificates that are stored in the key database. The information displayed includes, among others, the label, issuer/subject name, the version number of the certificate, the key size for the public/private key pair, and the expiration date.

To list information about certificates that contain private keys, from the Key Management Menu (see Figure 11 on page 262) select 1, (Manage keys and certificates). This displays the Key and Certificate List.

```
Key and Certificate List

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Test Server Cert
2 - Server Cert

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list): 2 <enter>
====>
```

*Figure 21. Key and Certificate List*

Select the number corresponding to the label for which you would like to display certificate/key information. The Key and Certificate Menu for the label you chose displays next (see Figure 22).

```
Key and Certificate Menu

Label: Server Cert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label

0 - Exit program

Enter option number (press ENTER to return to previous menu): 1 <enter>
====>
```

*Figure 22. Key and Certificate Menu*

On the Key and Certificate Menu, you could choose 1 to display certificate information. This accesses the Certificate Information menu (see Figure 23 on page 271):

```

Certificate Information

Label: Server Cert
Record ID: 13
Issuer Record ID: 13
Trusted: Yes
Version: 3
Serial number: 3c73c6d0000e8076
Issuer name: My Server Certificate
ID
IBM
Endicott
NY
US
Subject name: My Server Certificate
ID
IBM
Endicott
NY
US
Effective date: 2002/02/20
Expiration date: 2002/10/22
Public key algorithm: rsaEncryption
Public key size: 1024
Signature algorithm: sha1WithRsaEncryption
Issuer unique ID: None
Subject unique ID: None
Number of extensions: 4

Enter 1 to display extensions, 0 to return to menu: 1 <enter>
===>
```

Figure 23. Certificate Information

From the Certificate Information screen, you can also enter 1 to display certificate extensions:

```

Certificate Extensions List

1 - subjectKeyIdentifier
2 - authorityKeyIdentifier
3 - keyUsage (critical)
4 - basicConstraints (critical)

Enter extension number (press ENTER to return to previous menu): 3 <enter>
===>
```

Figure 24. Certificate Extensions List

Enter 3 on the Certificate Extensions List to show key usage information:

```

Certificate signature
CRL signature

Press ENTER to continue.
===>
```

Figure 25. Key Usage Information

To display key information, from the Key and Certificate Menu, choose 2, Show Key Information. This accesses the Key Information menu (see Figure 26):

```
Key Information
Label: Server Cert
Record ID: 13
Issuer Record ID: 13
Default key: Yes
Private key algorithm: rsaEncryption
Private key size: 1024
Subject name: My Server Certificate
              ID
              IBM
              Endicott
              NY
              US
Press ENTER to continue.
===>
```

Figure 26. Key Information menu

## Marking a Certificate (and Private Key) as the Default Certificate for the Key Database

Once a certificate has been added to the key database through either a certificate request or as a self-signed certificate, it can be marked as the default certificate for the key database. Marking a certificate as the default certificate allows it to be used by the programs calling the System SSL APIs without having to explicitly supply the certificate's label.

To mark a certificate as the default certificate for the key database, from the Key Management Menu (see Figure 11 on page 262), choose 1, (Manage keys and certificates), and on the Key and Certificate List (see Figure 21 on page 270, choose the label number you want to work with. The Key and Certificate Menu displays:

```
Key and Certificate Menu
Label: My Server Certificate
1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
0 - Exit program
Enter option number (press ENTER to return to previous menu): 3 <enter>
Default key set.
Press ENTER to continue.
===>
```

Figure 27. Marking a Certificate (and Private Key) as the Default Certificate

Choose 3 to set the certificate and private key as the default certificate for the key database.

### Copying a Certificate (and Private Key) to a Different Key Database

Once you have populated your key database file with certificates, it may be necessary for you to transfer a certificate to another key database on your system or a remote system. This transfer maybe necessary for the following reasons:

- The remote system or key database requires the signing certificate to be in its key database file for validation purposes. The certificate does not need to contain the private key information. These certificates are normally certificate authority (CA) certificates but may also be a self-signed certificate.
- The server or client certificate is being used by another application in a separate key database file

**Copying a Certificate Without its Private Key:** To copy a certificate to a different key database format or to a different system without its private key (certificate validation), from the Key Management Menu, select 1 - Manage keys and certificates to display the Key and Certificate List menu . Find the label of the certificate to be copied and enter the number associated with the label. In the Key and Certificate Menu, enter option 6 to export the certificate to a file. The Export File Format menu appears:

```
Export File Format

1 - Binary ASN.1 DER
2 - Base64 ASN.1 DER
3 - Binary PKCS #7
4 - Base64 PKCS #7

Select export format (press ENTER to return to menu): 1 <enter>
Enter export file name (press ENTER to return to menu): expfile.der <enter>

Certificate exported.

Press ENTER to continue.
===>
```

Figure 28. Copying a Certificate Without its Private Key

You are then prompted for what file format you would like for the exported certificate information.

The file format is determined by the support on the receiving system. When the receiving system implementation is z/OS System SSL V1R2 or earlier, the selected format **must** be one of the ASN.1 DER formats.

After selecting the export format, you will be asked for a file name. You then will receive a message indicating that the certificate was exported. You can now transfer this file to the system and import the certificate into the key database file. If copying to a remote system, this file can now be transferred (in binary) to the remote system. For information on receiving the certificate into the key database file, see “Importing a Certificate from a File as a Trusted CA Certificate” on page 277). Upon successfully receiving the certificate, the certificate can now be used to validate the SSL’s partner certificate. When the partner is the client, this means that the client can now validate the server’s certificate and if the partner is the server, the server can validate the client’s certificate when client authentication is requested.

You will also need to determine whether the certificate should be marked as the database’s default certificate. Setting the certificate as the default certificate allows the certificate to be used by the SSL APIs without having to specify its label. For more information on setting the default certificate, see “Marking a Certificate (and Private Key) as the Default Certificate for the Key Database” on page 272.

**Copying a Certificate with its Private Key:** To copy a certificate to a different key database format or to a different system with its private key, the certificate must be exported to a PKCS #12 formatted file. PKCS

#12 files are password-protected to allow encryption of the private key information. From the Key Management Menu, select 1 - Manage keys and certificates to display the Key and Certificate List menu. Find the label of the certificate to be copied and enter the number associated with the label. In the Key and Certificate Menu, enter option 7 to export the certificate and private key to a file.

The Export File Format menu appears:

```
Export File Format

1 - Binary PKCS #12 Version 1
2 - Base64 PKCS #12 Version 1
3 - Binary PKCS #12 Version 3
4 - Base64 PKCS #12 Version 3

Select export format (press ENTER to return to menu): 1 <enter>
Enter export file name (press ENTER to return to menu): expfile.p12

Certificate exported.

Press ENTER to continue.
===>
```

Figure 29. Copying a Certificate and Private key to a Different Key Database

You will then be prompted for what file format you would like for the exported certificate information.

The file format is determined by the support on the receiving system. In most cases the format to be used is Binary PKCS #12 Version 3. When the receiving system implementation is z/OS System SSL V1R2 or earlier, the selected format **must** be Binary PKCS #12 Version 1.

After selecting the export format, you will be asked for a file name and password. You then will receive a message indicating that the certificate was exported. You can now transfer this file to the system and import the certificate into the key database file. If copying to a remote system, this file can now be transferred (in binary) to the remote system. For information on receiving the certificate into the key database file, see "Importing a Certificate from a File with its Private Key" on page 278). Upon successfully receiving the certificate, the certificate can now be used to identify the program. For example, the certificate can be used as the SSL server program's certificate or it can be used as the SSL client program's certificate.

**Copying a Certificate with its Private Key to a Key Database on the Same System:** To copy a certificate and its private key from one key database to another key database on the same system, you will need to know the target key database file name and password. From the Key Management Menu, select 1 - Manage keys and certificates to display the Key and Certificate List menu. Find the label of the certificate to be copied and enter the number associated with the label. From the Key and Certificate Menu, enter 5 to copy a certificate and key to another database:

```
Key and Certificate Menu

Label: newimp

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label

0 - Exit program

Enter option number (press Enter to return to previous menu): 5 <enter>
Enter key database name (press Enter to return to previous menu): target.kdb <enter>
Enter database password (press Enter to return to previous menu): <enter password>
Record copied.

Press ENTER to continue.
====>
```

Figure 30. Copying a Certificate with its Private Key to a Key Database on the Same System

You will then be prompted for the target key database name, and the target key database password. Once the certificate is copied to the other key database file, you will receive a message indicating that the certificate has been successfully copied.

### Removing a Certificate (and Private Key) from a Key Database

You may want to remove a certificate from the key database. For example, you may want to remove a certificate after a certificate has expired and is no longer useful. Also, you may want to remove a certificate after a certificate has been exported to a different key database and is no longer needed as part of this key database.

**Caution:** Once you remove a certificate/private key pair from a key database, you cannot recover it unless it has previously been stored somewhere else (another key database file, a PKCS #12 file for certificate/private key pairs, or a DER-encoded or Base64-encoded file for just certificates). Be sure you no longer require the certificate (and private key if one is associated with the certificate in the key database) before you remove it from the key database.

From the Key Management Menu, select 1 - Manage keys and certificates to display the Key and Certificate List menu. Find the label chosen when creating (or receiving) the certificate and enter the number associated with the label. In the Key and Certificate Menu (see Figure 31 on page 276), choose 8 to delete the certificate and key:

Key and Certificate Menu

Label: newimp

- 1 - Show certificate information
- 2 - Show key information
- 3 - Set key as default
- 4 - Set certificate trust status
- 5 - Copy certificate and key to another database
- 6 - Export certificate to a file
- 7 - Export certificate and key to a file
- 8 - Delete certificate and key
- 9 - Change label
  
- 0 - Exit program

Enter option number (press ENTER to return to previous menu): **8** <enter>

Enter 1 to confirm delete, 0 to cancel delete: **1** <enter>

Record deleted.

Press ENTER to continue.

==>

Figure 31. Delete Certificate and Key

Enter 1 to confirm the deletion of the certificate and key. A message appears, confirming that the record has been deleted. Once the certificate has been removed from the key database, it can no longer be used for identification or verification purposes by the System SSL APIs during SSL handshake processing.

### Changing a Certificate Label

From the Key Management Menu, select 1 - Manage keys and certificates to display the Key and Certificate List menu. Find the label chosen when creating (or receiving) the certificate and enter the number associated with the label. In the Key and Certificate Menu (see Figure 32 on page 277), choose 9 to change the label:

Key and Certificate Menu

Label: cacert

- 1 - Show certificate information
- 2 - Show key information
- 3 - Set key as default
- 4 - Set certificate trust status
- 5 - Copy certificate and key to another database
- 6 - Export certificate to a file
- 7 - Export certificate and key to a file
- 8 - Delete certificate and key
- 9 - Change label
  
- 0 - Exit program

Enter option number (press ENTER to return to previous menu): **9** <enter>

Enter label (press ENTER to return to menu): **cacert2** <enter>

Label changed.

Press ENTER to continue.

==>

Figure 32. Changing a Certificate Label

Enter the new label name and press Enter. A message confirms that the label name has been changed.

## Importing a Certificate from a File as a Trusted CA Certificate

If you are using a certificate authority for generating your certificates that is not one of the default certificate authorities for which certificates are already stored in the key database, then you must import the certificate authority's certificate into your key database file before you use the System SSL APIs. If you are using **client authentication**, then the CA certificate must be imported into the key database of the server program. The client program's key database file must have the CA certificate imported regardless of whether or not the SSL connection uses **client authentication**.

If you are using a self-signed certificate as the SSL server program's certificate and your SSL client program is also using the System SSL APIs, then you must import the server's self-signed certificate into the client program's key database file without a private key.

If you are using a self-signed certificate as the SSL client program's certificate and your SSL server program is also using the System SSL APIs with client authentication requested, then you must import the client's self-signed certificate into the server program's key database file without a private key.

A number of well-known certificate authority's (CA) certificates are stored in the key database when the key database is created. Figure 33 and Figure 34 on page 278 contain lists of CAs for which certificates are stored on key database creation:

```
Certificate List

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - VeriSign Class 1 Public Primary CA
2 - VeriSign Class 2 Public Primary CA
3 - VeriSign Class 3 Public Primary CA
4 - RSA Secure Server CA
5 - Thawte Server CA
6 - Thawte Premium Server CA
7 - Thawte Personal Basic CA
8 - Thawte Personal Freemail CA
9 - Thawte Personal Premium CA

0 - Return to selection menu

Enter label number (ENTER for more labels, p for previous list):
===>
```

Figure 33. Certificate List (part 1)

```
Certificate List

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - VeriSign Class 1 Individual Subscriber-Persona Not Validated
2 - VeriSign Class 2 Individual Subscriber-Persona Not Validated
3 - VeriSign Class 3 Individual Subscriber-Persona Not Validated

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
===>
```

Figure 34. Certificate List (part 2)

To import a certificate without a private key into your key database file, first get the certificate in a file in the z/OS HFS with the file in either Base64-encoded, Binary encoded or PKCS #7 format. From the Key Management Menu enter 7 to import a certificate:

```
Key Management Menu

Database: /home/sufw11/ssl_cmd/mykey.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 7 <enter>
Enter import file name (press ENTER to return to menu): cert.arm <enter>
Enter label (press ENTER to return to menu): cacert2 <enter>

Certificate imported.

Press ENTER to continue.
====>
```

Figure 35. Importing a Certificate from a File

You will be prompted to enter the certificate file name and unique label to be assigned to the certificate.

Once the certificate is imported, you will receive a message indicating the import was successful. The certificate is treated as "trusted" so that it can be used in verifying incoming certificates. For a program acting as an SSL server, this certificate is used during the verification of a client's certificate. For a program acting as an SSL client, this certificate is used to verify the server's certificate which is sent to the client during SSL handshake processing.

## Importing a Certificate from a File with its Private Key

To store a certificate into a different key database format or to a different system with its private key, the certificate must be exported from the source system into a PKCS #12 format file (See Copying a Certificate with its Private Key for more information). PKCS #12 files are password-protected to allow encryption of the private key information. From the Key Management Menu, enter 8 to import a certificate and a private key:

```
Key Management Menu

Database: /home/sufw11/ssl_cmd/anne.kdb

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive certificate issued for your request
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 8 <enter>
Enter import file name (press ENTER to return to menu): cert.p12 <enter>
Enter import file password (press ENTER to return to menu): <enter password>
Enter label (press ENTER to return to menu): newcert <enter>

Certificate and key imported.

Press ENTER to continue.
===>
```

Figure 36. Importing a Certificate and Private Key from a File

You will be prompted to enter the certificate file name, password and a unique label to be assigned to the certificate.

Once the certificate is imported, you will receive a message indicating that import was successful. The next step is to determine whether the certificate should be marked as the database's default certificate. Setting the certificate as the default certificate allows the certificate to be used by the SSL APIs without having to specify its label. For more information on setting the default certificate, see "Marking a Certificate (and Private Key) as the Default Certificate for the Key Database" on page 272).

## Using gskkyman to be Your Own Certificate Authority (CA)

The **gskkyman** command provides the capability for you to act as your own Certificate Authority (CA). Being your own CA allows you to sign your own or anyone else's certificate requests. This is very handy if you only need the certificates within your private Web network and not for outside Internet commerce. Clients that will be interacting with your Web server must have browsers that can receive your CA certificate and designate you as a trusted CA. Before signing a certificate for a client or server, you need to make sure that the requestor has a legitimate claim to request the certificate. After you have verified the claim, you can create a signed certificate.

To be your own CA in a Web network, you must create a CA database and self-signed CA certificate using **gskkyman**. A server or client that wishes for you to sign a certificate must supply you with their certificate request. After signing the certificate, the server or client must then download your CA certificate and the newly signed certificate. The CA-signed certificate must then be received into either the client or server key database.

The following example illustrates the steps from creating the CA database to having a CA-signed certificate to allow secure communication between a client and a server. This example reflects the steps followed when the CA is on a different system or is a different user than the issuer of the certificate

request. If a single user is performing both the creation of the certificate request and the signing, sending of the files are not needed.

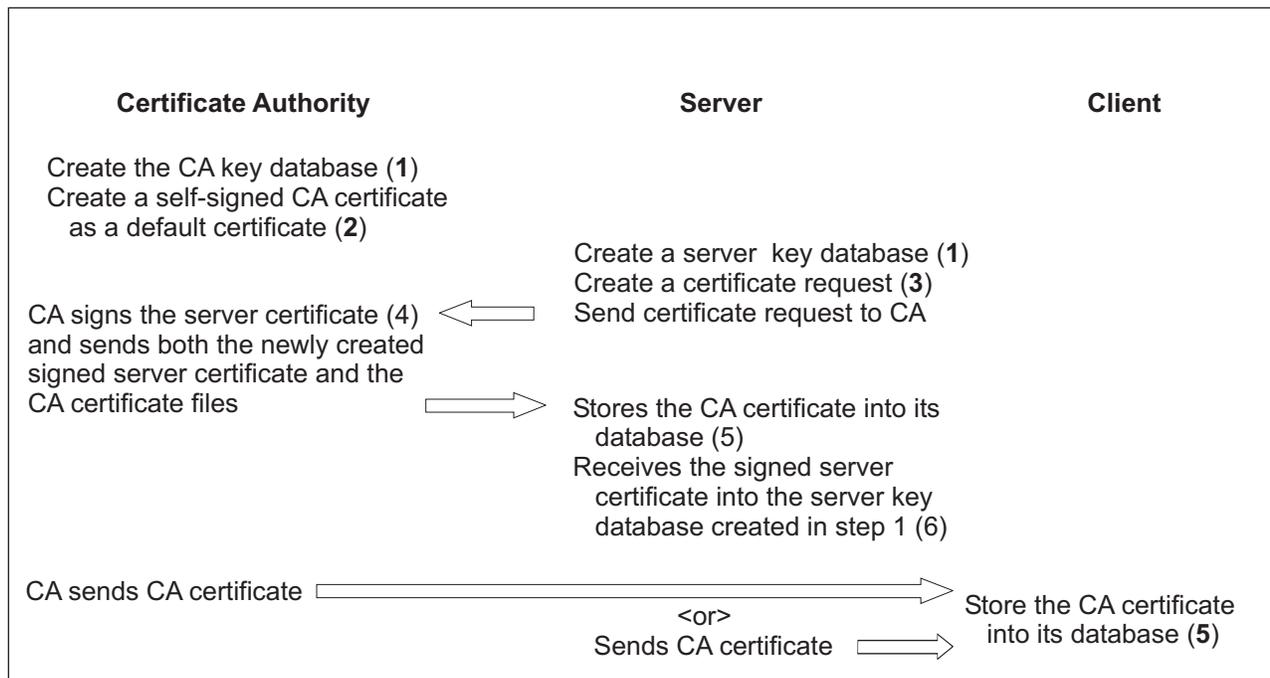


Figure 37. Being Your Own CA in a Web Network

1. To create a key database, see “Creating, Opening and Deleting a Key Database File” on page 260.
2. To create a Certificate Authority (CA) certificate, you need to follow the same procedure as creating a self-signed server certificate except the certificate type is CA. See “Creating a Self-Signed Server or Client Certificate” on page 264.
3. To create a certificate request, see “Creating a Certificate Request and Processing the Signed Request” on page 266.
4. To sign a certificate, the **gskkyman** command must be issued using command-line options. The **gskkyman** command must be issued with the following parameters:
 

```
gskkyman -g -x num-of-valid-days -cr certificate-request-file-name
      -ct signed-certificate-file-name -k CA-key-database-file-name
```

**Note:** The CA certificate start/end dates must be before/after those of the requested certificate.

For example, the following command will allow you to sign a request certificate and allow the certificate to be valid for 360 days.

```
gskkyman -g -x 360 -cr server_request.arm -ct server_signed_cert.arm -k CA.kdb
```

**Notes:**

- a. The valid certificate lifetime range is between 1 and 9999 days. The certificate end date will be set to the end date for the CA certificate if the requested certificate lifetime exceeds the CA certificate lifetime.
  - b. The signed certificate will be an end user certificate unless the **-ca** option is specified.
  - c. The filename specified on the **-ct** option is created for you by the utility, and is the actual signed server certificate file referred to in step 6 of the example.
5. To store a CA certificate, see “Importing a Certificate from a File as a Trusted CA Certificate” on page 277

6. To receive a signed certificate, see Figure 20 on page 269 (Receiving a Certificate Issued for your Request). Depending upon the SSL application, you may need to either send the CA certificate to the client, or the server application may actually present the certificate to the client for them during SSL session setup.

## Migrating Key Database Files to RACF Key Rings

If you need to migrate keys and certificates stored in an existing z/OS HFS key database into a RACF key ring, follow these steps:

1. Export the certificate/private key to a password protected PKCS#12 file using **gskkyman**. Refer to “Copying a Certificate with its Private Key” on page 273 for details on the steps for exporting certificates/private keys to a PKCS#12 file.
2. Copy the newly created PKCS#12 file to a z/OS dataset.
3. Use the RACDCERT command with the ADD operand to define a certificate/private key. The dataset name created in step 2 contains the certificate.
4. Use the RACDCERT command with the ADDRING operand to create a new key ring in RACF. Use the RACDCERT command with the CONNECT operand to add the certificate/private key to one or more existing RACF key rings.

---

## Example Tasks Performed by the gskkyman Command in Command Mode

Command mode is entered when the gskkyman command is entered with parameters. The requested database function will be performed and then the command will exit.

- Store the database password in the stash file

```
gskkyman -s -k filename
```

The database password is masked and written to the key stash file. The file name is the same as the key database file name but has an extension of '.sth'. You will be prompted for the key database file name if the '-k' option is not specified.

- Export a certificate and the associated private key

```
gskkyman -e -k filename -l label -p filename
```

The certificate and associated private key identified by the record label are exported to a file in PKCS #12 Version 3 format using strong encryption. The default key will be exported if the '-l' option is not specified. You will be prompted for the key database file name if the '-k' option is not specified. You will be prompted for the export file name if the '-p' option is not specified.

- Import a certificate and associated private key

```
gskkyman -i -k filename -l label -p filename
```

A certificate and associated private key are imported from a file in PKCS #12 format. You will be prompted for the label if the '-l' option is not specified. You will be prompted for the key database file name if the '-k' option is not specified. You will be prompted for the import file name if the '-p' option is not specified.

- Create a signed certificate for a certificate request

```
gskkyman -g -x days -cr filename -ct filename -k filename -l label -ca
```

The certificate request identified by the -cr parameter is processed and a signed certificate is created and written to the certificate file identified by the -ct parameter. The -x parameter specifies the number of days until the certificate expires and defaults to 365 days. The certificate is signed using the default key if the -l parameter is not specified. You will be prompted for the key database file name if the '-k' option is not specified. You will be prompted for the certificate request file name if the '-cr' option is not specified. You will be prompted for the signed certificate file name if the '-ct' option is not specified.

The signed certificate will be an end user certificate unless the -ca option is specified. A certification authority certificate will have basic constraints and key usage extensions which allow the certificate to be used to sign other certificates and certificate revocation lists. An end user certificate will have basic constraints and key usage extensions which allow the certificate to be used for authentication, digital signatures, and data encryption (a DSA key cannot be used for data encryption).

Any certificate can be used to sign the new certificate as long as the certificate has a private key, the basic constraints certificate extension (if present) has the CA indicator set, and the key usage certificate extension (if present) allows signing certificates. However, depending upon how the new certificate is subsequently used, it may fail the validation checking if the signing certificate is not a valid certification authority certificate.

---

## Chapter 11. SSL Started Task

The SSL started task (GSKSRVR) provides sysplex session cache support and dynamic trace support. The SSL started task is an optional component of System SSL and does not need to be configured and started in order to use System SSL.

The default home directory for the SSL started task is `/etc/gskssl/server`. A different home directory can be specified by changing the definition of the HOME environment variable in the GSKSRVR procedure. The SSL started task will read the `envar` file in the home directory to set the environment variables. This file is a variable-length file where each line consists of a variable name and variable value separated by '='. Trailing blanks are removed from the variable value. Blanks lines and lines beginning with '#' are ignored.

---

### GSKSRVR Environment Variables

The following environment variables are processed by the System SSL started task.

#### GSK\_LOCAL\_THREADS

Specifies the number of threads which will be used to handle program call requests from SSL applications running on the same system as the GSKSRVR started task. The default value is 5 and the minimum value is 2. The default of 5 will be used if a valid value is not specified.

#### GSK\_SIDCACHE\_SIZE

Specifies the size of the sysplex session cache in megabytes and is between 1 and 512 with a default of 20. The default of 20 will be used if a valid value is not specified.

#### GSK\_SIDCACHE\_TIMEOUT

Specifies the sysplex session cache entry timeout in minutes and is between 1 and 1440 with a default of 60. The default of 60 will be used if a valid value is not specified.

---

### Configuring the SSL Started Task

1. Create the home directory for the SSL started task (the default is `/etc/gskssl/server`)
2. Copy the sample `envar` file from `/usr/lpp/gskssl/examples/gsksrvr.envar` to `/etc/gskssl/server/envar` (change the directory name to match the home directory created above) and modify the LANG, TZ, and NLSPATH values to meet local installation requirements.
3. Copy the sample started procedure from `GSK.SGSKSAMP(GSKSRVR)` to `SYS1.PROCLIB(GSKSRVR)`
4. Create the GSKSRVR user and associate it with the GSKSRVR started procedure. Replace 'nnnnnn' in the ADDUSER command with a non-zero value which is not assigned to another user.

```
ADDUSER GSKSRVR DFLTGRP(SYS1) NOPASSWORD OMVS(UID(nnnnnn)) PROGRAM(/bin/sh) HOME(/etc/gskssl/server)

RDEFINE STARTED GSKSRVR.** STDATA(USER(GSKSRVR) GROUP(SYS1) TRUSTED)

SETROPTS RACLIST(STARTED) REFRESH
```
5. Ensure that the `GSK.SGSKLOAD` and `CEE.SCEERUN` datasets are APF-authorized and are either in the link list concatenation or are specified as a STEPLIB for the GSKSRVR procedure.
6. Optionally, set up a message processing exit to automatically start the GSKSRVR started task. The `GSK.SGSKSAMP(GSKMSGXT)` program is a sample message processing exit for this purpose. In order to activate the exit, add the following to the appropriate MPFLSTxx member in `SYS1.PARMLIB`.

```
BPXI004I,SUP(NO),USEREXIT(STARTSSL)
```

This will start GSKSRVR when OMVS initialization is complete, assuming the GSKMSGXT program was linked as STARTSSL and placed in a LNKLIST dataset.

7. Optionally, set up an automatic restart management (ARM) policy for the GSKSRVR started task if the default ARM policy values are not appropriate. The element type is SYSSSL and should be assigned to restart level 2. The element name is GSKSRVR\_sysname. For example, the element name for the

GSKSRVR started task on system DCESEC4 would be GSKSRVR\_DCESEC4. Since the normal operating mode is to run the GSKSRVR started task on each system in the sysplex, the GSKSRVR started task will register with ARM to be restarted only if the started task fails and not if the current system fails. The TERMTYPE parameter of the ARM policy can be used to override this registration if desired.

---

## Server Operator Commands

The following operator commands are supported by the System SSL server:

### **STOP GSKSRVR or P GSKSRVR**

Causes an orderly shutdown of the server.

### **MODIFY GSKSRVR,parameters or F GSKSRVR,parameters**

Causes a command to be executed by the server. Some parameters are:

#### **DISPLAY CRYPTO**

Displays the available encryption algorithms, whether hardware cryptographic support is available, and the maximum encryption key size. 'N/A' will be displayed if the encryption algorithm is not available due to export restrictions.

This command can be abbreviated as 'D CRYPTO'

#### **DISPLAY LEVEL**

Displays the current System SSL service level.

This command can be abbreviated as 'D LEVEL'

#### **DISPLAY SIDCACHE**

Displays the current and maximum data space sizes in megabytes followed by the session cache users and the number of cache entries for each user. The count will include expired cache entries until they are removed from the cache during an update to the hash list containing the expired entry. Each GSKSRVR started task maintains its own session cache for sessions created on that system. The 'DISPLAY SIDCACHE' command must be issued for each started task to display the cache entries for the entire sysplex. This can be done by issuing 'RO \*ALL,F GSKSRVR,D SIDCACHE'.

This command can be abbreviated as 'D SIDCACHE'

#### **DISPLAY XCF**

Displays the status of all instances of the GSKSRVR started task in the sysplex.

This command can be abbreviated as 'D XCF'

**STOP** Causes an orderly shutdown of the server. This is the same as entering the "STOP GSKSRVR" command.

#### **TRACE OFF**

Turns off tracing for the System SSL started task.

#### **TRACE ON,level**

Turns on tracing for the System SSL started task. The trace output is written to the file specified by the GSK\_TRACE\_FILE environment variable or to the default trace file if the GSK\_TRACE\_FILE environment variable is not defined. The level value specifies the trace level. Refer to the descriptions of the GSK\_TRACE and GSK\_TRACE\_FILE environment variables for more information about SSL tracing.

---

## Sysplex Session Cache Support

The sysplex session cache support makes SSL server session information available across the sysplex. An SSL session established with a server on one system in the sysplex can be resumed using a server on another system in the sysplex as long as the SSL client presents the session identifier obtained for the first session when initiating the second session. SSL V3 and TLS V1 server session information can be stored in the sysplex session cache while SSL V2 server session information and all client session information is stored only in the SSL cache for the application process.

In order to use the sysplex session cache, each system in the sysplex must be using the same external security manager (for example, z/OS Security Server RACF) and a userid on one system in the sysplex must represent the same user on all other systems in the sysplex (that is, userid ZED on System A has the same access rights as userid ZED on System B). The external security manager must support the RACROUTE REQUEST=EXTRACT,TYPE=ENVRXTR and RACROUTE REQUEST=FASTAUTH functions.

The sysplex session cache must be enabled for each application server that is to use the support. This can be done by defining the GSK\_SYSPLEX\_SIDCACHE environment variable or by calling the **gsk\_attribute\_set\_enum()** routine to set the GSK\_SYSPLEX\_SIDCACHE attribute. The session information for each new SSL V3 or TLS V1 session created by the SSL server will then be stored in the sysplex session cache and can be referenced by other SSL servers in the sysplex. The RACF user associated with the SSL server becomes the owner of the session information. Any SSL server running with the same RACF user can access the session information. SSL servers running with a different RACF user can access the session information if they have at least READ access to the GSK.SIDCACHE.<owner> profile in the FACILITY class.

For example, session information created by RACF user APPLSRV1 can be accessed by RACF user APPLSRV2 if APPLSRV2 has READ access to the GSK.SIDCACHE.APPLSRV1 profile in the FACILITY class. The following RACF commands grant this access:

```
RDEFINE FACILITY GSK.SIDCACHE.APPLSRV1 UACC(NONE)
PERMIT GSK.SIDCACHE.APPLSRV1 CLASS(FACILITY) ID(APPLSRV2) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

---

## Component Trace Support

For information about component trace support, see “Component Trace Support” on page 288.



---

## Chapter 12. Obtaining Diagnostic Information

All of the information and techniques described in this chapter are for use primarily by IBM service personnel in determining the cause of a System SSL problem. If you encounter a problem and call the IBM Support Center, you may be asked to obtain trace information or enable one or more of the diagnostic messages described below.

Any environment variables described in this chapter are usually set via the UNIX System Services **export** shell command. For usage information on this command, see the *z/OS: UNIX System Services Command Reference*, SA22-7802. For information on setting environment variables outside of the shell, refer to the *z/OS: C/C++ Programming Guide*, SC09-4765 and the *z/OS: Language Environment Programming Guide*, SA22-7561.

**The facilities described below are not intended for use in a production environment and are for diagnostic purposes only.**

---

### Obtaining System SSL Trace Information

You can enable the System SSL trace by using the environment variable `GSK_TRACE_FILE` to specify the name of the trace file, and the `GSK_TRACE` environment variable to set the trace level. A single trace file is created, and there is no limit on the size of the trace file.

In order to create a readable copy of the trace information, use the System SSL **gsktrace** command as follows:

```
gsktrace input_trace_file > output_trace_file
```

### Capturing Trace Data Through Environment Variables

In order to capture trace information using environment variables, the trace environment variables **GSK\_TRACE** and **GSK\_TRACE\_FILE** must be exported prior to the start of the SSL application.

- **GSK\_TRACE**

Specifies a bit mask enabling System SSL trace options. No trace option is enabled if the bit mask is 0 and all trace options are enabled if the bit mask is 0xffff. The bit mask can be specified as a decimal (nnn), octal (0nnnn) or hexadecimal (0xhh) value.

The following trace options are available:

- 0x01 = Trace function entry
- 0x02 = Trace function exit
- 0x04 = Trace errors
- 0x08 = Include informational messages
- 0x10 = Include EBCDIC data dumps
- 0x20 = Include ASCII data dumps

- **GSK\_TRACE\_FILE**

Specifies the name of the trace file and defaults to `/tmp/gskssl.%.trc`. The trace file is not used if the **GSK\_TRACE** environment variable is not defined or is set to 0.

The current process identifier is included as part of the trace file name when the name contains a percent sign (%). For example, if `GSK_TRACE_FILE` is set to `/tmp/gskssl.%.trc` and the current process identifier is 247, then the trace file name will be `/tmp/gskssl.247.trc`.

**Note:**

Care needs to be taken if the application being traced is multi-processed. If multiple processes write to the same trace file, file corruption may occur. To allow trace information to be obtained, the trace

## Obtaining Diagnostic Information

file name specified should contain a '%' character in the file name. This will allow the process identifier to be placed within the file name and each process to write to its own trace file.

It is recommended that if the default trace file value is not being used, the trace file name always contain a '%' character. This will eliminate the need to know if the application being traced is multi-processed or not.

Once the trace file is produced, it must be formatted. To format the file, use the System SSL gsktrace command as follows:

```
gsktrace input_trace_file > output_trace_file
```

---

## Component Trace Support

The System SSL started task provides component trace support for any SSL application running on the same system as the GSKSRVR started task. The trace records can be written to a trace external writer or they can be kept in an in-storage trace buffer which is part of the GSKSRVR address space. IPCS is used to format and display the trace records from either a trace dataset or an SVC dump of the GSKSRVR address space.

Refer to *MVS Diagnosis: Tools and Service Aids* for more information on setting up and using component trace. Refer to *MVS System Commands* for more information on the TRACE command. Refer to *MVS IPCS User's Guide* for more information on using IPCS to view a component trace.

---

## Capturing Component Trace Data

The component trace can be started before the job to be traced is started or while the job is running. The trace will be active for the first instance of the job. For example, if the same job name is used for multiple jobs, only the first job with that name will be traced. Subsequent jobs with the same name will not be traced unless the component trace is stopped and then restarted.

A trace external writer is required if the trace records are to be written to a dataset. A sample started procedure is shipped as GSK.SGSKSAMP(GSKWTR). Copy this procedure to SYS1.PROCLIB(GSKWTR) and modify as necessary to meet your installation requirements. The following MVS operator command will start the trace external writer:

```
TRACE CT,WTRSTART=GSKWTR
```

A single SSL component trace may be active at a time and the trace can include from 1 to 16 separate jobs. The trace buffer size must be between 64K and 512K and will default to 64K. The OPTIONS parameter specifies the SSL trace level as 'LEVEL=mask' where 'mask' specifies a bit mask enabling System SSL trace options. No trace option is enabled if the bit mask is 0 and all trace options are enabled if the bit mask is 0xffff. The bit mask can be specified as a decimal (nnn), octal (0nnnn) or hexadecimal (0xhh) value. The SSL trace level will be set to 15 if no trace options are specified.

The following trace options are available:

- 0x01 = Trace function entry
- 0x02 = Trace function exit
- 0x04 = Trace errors
- 0x08 = Include informational messages
- 0x10 = Include EBCDIC data dumps
- 0x20 = Include ASCII data dumps

For example, to start an SSL component trace for jobs CS390IP and DB1G which includes all non-dump trace entries and writes the trace records using the GSKWTR trace writer:

```
TRACE CT,ON,COMP=GSKSRVR  
R n,JOBNAME=(CS390IP,DB1G),OPTIONS=(LEVEL=15),WTR=GSKWTR,END
```

## Obtaining Diagnostic Information

The following commands will stop the SSL component trace and close the trace writer dataset:

```
TRACE CT,OFF,COMP=GSKSRVR  
TRACE CT,WTRSTOP=GSKWTR
```

System SSL does not require a default trace member in SYS1.PARMLIB since SSL component trace is not activated until the operator enters the TRACE command. SYS1.PARMLIB members can be created for frequently used trace commands and the member name can then be specified on the TRACE command to avoid the operator prompt for trace options.

---

### Displaying the Trace Data

The trace records are displayed using the IPCS CTRACE command.

The CTRACE ENTIDLIST parameter specifies the trace entries to be included in the display. The trace entry type is the same as the SSL trace level. For example, SSL function entry trace records have entry type 1, SSL function exit trace records have entry type 2, SSL error records have entry type 4, etc. All trace entries will be included if the ENTIDLIST parameter is not specified.

The CTRACE OPTIONS parameter specifies additional filtering for the trace records. The JOB(name), PID(hexid), and TID(hexid) options can be specified to filter the trace entries based on job name, process identifier, or thread identifier. All trace entries will be included if the OPTIONS parameter is not specified.

Note that the JOBNAME parameter on the CTRACE command is used to select the address space in a dump. Since the address space is always the GSKSRVR address space, this parameter cannot be used to filter the trace entries. Instead, you must use the OPTIONS((JOB(name))) parameter to select the component trace entries for a specific job.

For example, to display SSL function entry and SSL function exit trace records for job KRBSRV48 thread 6:

```
IPCS CTRACE COMP(GSKSRVR) ENTIDLIST(1,2) OPTIONS((JOB(KRBSRV48),TID(6))) FULL
```

A range can be specified for the entry identifiers. For example, to display just the non-dump trace records:

```
IPCS CTRACE COMP(GSKSRVR) ENTIDLIST(1:15) FULL
```

## Obtaining Diagnostic Information

---

## Chapter 13. Messages and Codes

This chapter contains information for the various forms of messages and codes you may encounter:

- SSL Function Return Codes
- Deprecated SSL Function Return Codes
- ASN.1 Status Codes (014CExxx)
- CMS Status Codes (03353xxx)
- SSL Started Task Messages (GSK01nnn)
- Utility Messages (GSK00nnn)

---

### SSL Function Return Codes

This section describes the SSL function return codes.

---

#### 1 Handle is not valid.

**Explanation:** The environment or SSL handle specified on a System SSL function call is not valid.

**User Response:** Call the `gsk_environment_open()` function to create an environment handle or the `gsk_secure_socket_open()` function to create an SSL handle.

---

#### 3 An internal error has occurred.

**Explanation:** The System SSL runtime library has detected an internal processing error.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

#### 4 Insufficient storage is available

**Explanation:** The System SSL runtime library is unable to obtain storage for an internal control block.

**User Response:** Increase the storage available to the application and then retry the failing operation.

---

#### 5 Handle is in the incorrect state.

**Explanation:** The SSL handle is in the incorrect state for the requested operation.

**User Response:** Correct the application to request SSL functions in the proper sequence.

---

#### 6 Key label is not found.

**Explanation:** The requested key label is not found in the key database or SAF key ring.

**User Response:** Specify a label that exists in the key database or SAF key ring.

---

#### 7 No certificates available.

**Explanation:** The key database or SAF key ring does not contain any certificates.

**User Response:** Add the user certificate and any necessary certification authority certificates to the key database or SAF key ring.

---

#### 8 Certificate validation error.

**Explanation:** An error is detected while validating a certificate. This error can occur if a root CA certificate is not found in the key database or SAF keyring or if the certificate is not marked as a trusted certificate.

**User Response:** Verify that the root CA certificate is in the key database or SAF keyring and is marked as trusted. Check all certificates in the certification chain and verify that they are trusted and are not expired. Collect a System SSL trace containing the error and then contact your service representative if the problem persists.

---

#### 9 Cryptographic processing error.

**Explanation:** An error is detected by a cryptographic function.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

#### 10 ASN processing error.

**Explanation:** An error is detected while processing a certificate field.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**11 LDAP processing error.**

**Explanation:** An error is detected while setting up the LDAP environment or retrieving an LDAP directory entry.

**User Response:** Ensure that the LDAP server is running and that there are no network errors. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**12 An unexpected error has occurred.**

**Explanation:** An unexpected error is detected by the System SSL runtime.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**102 Error detected while reading key database or SAF key ring.**

**Explanation:** An error is detected while reading the key database or retrieving entries on the SAF key ring.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**103 Incorrect key database record format.**

**Explanation:** The record format for a key database entry is not correct. This error can occur if the name of a request database is provided instead of the name of a key database.

**User Response:** Ensure that the correct database name is used. Collect a System SSL trace containing a dump of the keyfile entry and then contact your service representative if the error persists.

---

**106 Incorrect key database password.**

**Explanation:** The System SSL runtime is unable to decrypt a key database entry. Either the supplied database password is incorrect or the database is damaged.

**User Response:** Ensure that the correct key database password is used. Recreate the database if the error persists.

---

**109 No certification authority certificates.**

**Explanation:** The key database or SAF key ring does not contain any valid certification authority certificates. The SSL runtime needs at least one CA or self-signed certificate in order to perform client authentication.

**User Response:** Add the necessary certificates to the key database or SAF key ring and ensure that existing

certificates are valid, have not expired and are marked as trusted certificates.

---

**201 No key database password supplied.**

**Explanation:** A password stash file is specified but the SSL runtime is unable to read the password from the stash file.

**User Response:** Verify that the password stash file exists and is accessible to the application. Recreate the password stash file if the error persists.

---

**202 Error detected while opening the key database.**

**Explanation:** An error is detected while opening the key database or the SAF key ring. This error can occur if no name is supplied or the database or key ring does not exist.

**User Response:** Verify that the key database or SAF key ring exists and is accessible by the application. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**203 Unable to generate temporary key pair**

**Explanation:** An error is detected while generating a temporary key pair.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**204 Key database password is expired.**

**Explanation:** The key database password is expired.

**User Response:** Use the gskkyman command to assign a new password for the key database.

---

**302 Connection is active.**

**Explanation:** An SSL secure connection operation cannot be completed due to an active request for the connection.

**User Response:** Retry the failing request when the currently active request has completed.

---

**401 Certificate is expired or is not valid yet.**

**Explanation:** The current time is either before the certificate start time or after the certificate end time.

**User Response:** Obtain a new certificate if the certificate is expired or wait until the certificate becomes valid if it is not valid yet.

---

---

**402 No SSL cipher specifications.**

**Explanation:** The client and server cipher specifications do not contain at least one value in common. This error can also occur if no SSL protocols are enabled or if all of the enabled protocols have empty cipher specifications.

**User Response:** Ensure that the client and the server have at least one cipher specification in common.

---

**403 No certificate received from partner.**

**Explanation:** The required certificate was not received from the communication partner.

**User Response:** Ensure that the remote application is sending the certificate. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**405 Certificate format is not supported.**

**Explanation:** The certificate received from the communication partner is not supported by the current version of the System SSL runtime.

**User Response:** Collect a System SSL trace containing a dump containing the unsupported certificate and then contact your service representative.

---

**406 Error while reading or writing data.**

**Explanation:** An I/O error was reported while the System SSL runtime was reading or writing data.

**User Response:** Ensure that there are no network errors. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**407 Key label does not exist.**

**Explanation:** The supplied label or the default key is not found in the key database or the certificate is not trusted.

**User Response:** Supply a valid label or define a default key in the key database.

---

**408 Key database password is not correct.**

**Explanation:** The System SSL runtime is unable to decrypt a keyfile entry. Either the supplied keyfile password is incorrect or the keyfile is damaged.

**User Response:** Ensure that the correct keyfile password is used. Recreate the keyfile if the error persists.

---

---

**410 SSL message format is incorrect.**

**Explanation:** An incorrectly formatted SSL message is received from the communication partner.

**User Response:** Collect a System SSL trace containing a dump of the SSL message and then contact your service representative.

---

**411 Message authentication code is incorrect.**

**Explanation:** The message authentication code (MAC) for a message is not correct. This indicates the message was modified during transmission.

**User Response:** Collect a System SSL trace containing a dump of the message and then contact your service representative if the error persists.

---

**412 SSL protocol or certificate type is not supported.**

**Explanation:** The SSL handshake is not successful due to an unsupported protocol or certificate type. This error can occur if there is no enabled SSL protocol shared by both the client and the server.

**User Response:** Ensure that the desired SSL protocol is enabled on both the client and the server. Collect a System SSL trace containing a dump of the failing handshake and then contact your service representative if the problem persists.

---

**413 Certificate signature is incorrect.**

**Explanation:** The certificate signature is not correct for a certificate received from the communication partner.

**User Response:** Ensure that a valid certificate is being sent by the communication partner. Collect a System SSL trace containing a dump of the incorrect certificate and then contact your service representative if the error persists.

---

**414 Certificate is not valid.**

**Explanation:** Either the local certificate or the peer certificate is not valid.

**User Response:** Ensure that a valid certificate is being sent by the communication partner. Collect a System SSL trace containing a dump of the incorrect certificate and then contact your service representative if the error persists.

---

**415 SSL protocol violation.**

**Explanation:** The communication partner has violated the SSL protocol by sending a message out of sequence or by omitting a required field from a message.

---

**User Response:** Collect a System SSL trace and then contact your service representative.

---

**416 Permission denied.**

**Explanation:** The System SSL runtime is unable to access a file or system facility.

**User Response:** Ensure the application is authorized to access the file or facility. Collect a System SSL trace and then contact your service representative if the error persists.

---

**417 Self-signed certificate cannot be validated.**

**Explanation:** A self-signed certificate cannot be validated because it is not in the key database or SAF key ring.

**User Response:** Add the self-signed certificate to the key database or SAF key ring.

---

**420 Socket closed by remote partner.**

**Explanation:** The remote partner closed the socket. This error will also be reported if the remote partner has sent a close notification alert.

**User Response:** None.

---

**421 SSL V2 cipher is not valid.**

**Explanation:** The SSL V2 cipher is not valid.

**User Response:** Specify a valid cipher.

---

**422 SSL V3 cipher is not valid.**

**Explanation:** The SSL V3 cipher is not valid.

**User Response:** Specify a valid cipher.

---

**427 LDAP is not available.**

**Explanation:** The System SSL runtime is unable to access the LDAP server.

**User Response:** Ensure that the LDAP server is running and that there are no network problems. Collect a System SSL trace and then contact your service representative if the error persists.

---

**428 Key entry does not contain a private key.**

**Explanation:** The key entry does not contain a private key. This error can also occur if the private key is stored in ICSF and ICSF services are not available.

**User Response:** Specify a key entry containing a private key value. Ensure that the ICSF started task is running if the private key is stored in ICSF.

---

**429 SSL V2 header is not valid.**

**Explanation:** The received message does not start with a valid SSL V2 header. This error can occur if an SSL V3 client attempts to establish a secure connection with an SSL V2 server.

**User Response:** Enable the SSL V2 protocol on the client and then retry the request.

---

**431 Certificate is revoked.**

**Explanation:** The certificate has been revoked by the certification authority.

**User Response:** Obtain a new certificate.

---

**432 Session renegotiation is not allowed.**

**Explanation:** An attempt to renegotiate the session parameters for an active connection is rejected by the peer application.

**User Response:** SSL processing continues using the current session parameters.

---

**433 Key exceeds allowable export size.**

**Explanation:** The key size used for an export cipher suite exceeds the allowable maximum size. For RSA and DSA keys, the maximum export key size is 512 bits. If the certificate key is larger than 512 bits, the SSL runtime will use a temporary 512-bit key for the connection.

**User Response:** Collect a System SSL trace and then contact your service representative.

---

**434 Certificate key is not compatible with cipher suite.**

**Explanation:** The certificate key is not compatible with the negotiated cipher suite. The server certificate must have an RSA key while the client certificate may have an RSA or DSA key. This error can also occur if the client certificate has a DSA key but the server does not support DSA keys, the server key usage certificate extension does not allow key encipherment, or the client key usage certificate extension does not allow digital signature. For the 40-bit export ciphers, the server key usage certificate extension must allow digital signature.

**User Response:** Specify a certificate with the appropriate key type and key usage.

---

**435 Certification authority is unknown.**

**Explanation:** The key database does not contain a certificate for the certification authority.

**User Response:** Obtain the certificate for the certification authority and add it to the key database.

---

**436 Certificate revocation list cannot be processed.**

**Explanation:** A certificate revocation list (CRL) is not valid and cannot be processed.

**User Response:** Contact the certification authority and obtain a replacement CRL.

---

**437 Connection closed.**

**Explanation:** For `gsk_secure_socket_read()`, a close notification has been received from the peer application. For `gsk_secure_socket_write()`, a close notification has been sent to the peer application. A close notification is sent when the `gsk_secure_socket_shutdown()` routine is called or when a close notification is received from the peer application. Additional data may not be sent by the application after the close notification has been sent to the peer application.

**User Response:** None

---

**438 Internal error reported by remote partner.**

**Explanation:** The peer application has detected an internal error while performing an SSL operation and has sent an alert to close the secure connection.

**User Response:** Check the error log for the remote application to determine the nature of the processing error.

---

**439 Unknown alert received from remote partner.**

**Explanation:** The peer application has sent an alert message which is not recognized by the System SSL runtime.

**User Response:** Collect a System SSL trace and then contact your service representative.

---

**501 Buffer size is not valid.**

**Explanation:** The socket buffer or buffer size is not valid.

**User Response:** Specify a valid buffer and buffer size.

---

**502 Socket request would block.**

**Explanation:** The socket is in non-blocking mode and the socket request returned the EWOULDBLOCK error.

**User Response:** Retry the `gsk_secure_socket_read()` or `gsk_secure_socket_write()` request when the socket is ready to send or receive data.

---

---

**503 Socket read request would block.**

**Explanation:** A socket read request issued as part of an SSL handshake returned the EWOULDBLOCK error.

**User Response:** Retry the failing request when the socket is ready to receive data.

---

**504 Socket write request would block.**

**Explanation:** A socket write request issued as part of an SSL handshake return the EWOULDBLOCK error.

**User Response:** Retry the failing request when the socket is ready to send data.

---

**505 Record overflow.**

**Explanation:** An SSL protocol record has a plain text record length greater than 16384 or an encrypted text record length greater than 18432.

**User Response:** Ensure that data is not being corrupted during transmission. Obtain a System SSL trace containing a dump of the failing record and contact your service representative if the error persists.

---

**601 Protocol is not SSL V3 or TLS V1.**

**Explanation:** The requested function requires either the SSL V3 or the TLS V1 protocol.

**User Response:** Ensure that the correct protocol is in use before issuing the request.

---

**602 Function identifier is not valid.**

**Explanation:** The function identifier specified for `gsk_secure_socket_misc()` is not valid.

**User Response:** Specify a valid function identifier.

---

**701 Attribute identifier is not valid.**

**Explanation:** The attribute identifier is not valid.

**User Response:** Specify a valid attribute identifier.

---

---

## Deprecated SSL Function Return Codes

This section describes the deprecated SSL function return codes.

---

### 1 Error detected while reading key database or SAF key ring.

**Explanation:** An error is detected while reading the key database or retrieving entries on the SAF key ring.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

### 2 Error detected while opening the key database.

**Explanation:** An error is detected while opening the key database or the SAF key ring. This error can occur if no name is supplied or the database or key ring does not exist.

**User Response:** Verify that the key database or SAF key ring exists and is accessible by the application. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

### 3 Incorrect key database record format.

**Explanation:** The record format for a key database entry is not correct. This error can occur if the name of a request database is provided instead of the name of a key database.

**User Response:** Ensure that the correct database name is used. Collect a System SSL trace containing a dump of the keyfile entry and then contact your service representative if the error persists.

---

### 4 Key database password is not correct.

**Explanation:** The System SSL runtime is unable to decrypt a keyfile entry. Either the supplied keyfile password is incorrect or the keyfile is damaged.

**User Response:** Ensure that the correct keyfile password is used. Recreate the keyfile if the error persists.

---

### 9 Key label does not exist.

**Explanation:** The supplied label or the default key is not found in the key database or the certificate is not trusted.

**User Response:** Supply a valid label or define a default key in the key database.

---

### 12 Key label is not found.

**Explanation:** The requested key label is not found in the key database or SAF key ring.

**User Response:** Specify a label that exists in the key database or SAF key ring.

---

### 13 Duplicate subject names.

**Explanation:** The key database or SAF key ring contains multiple certificates with the same subject name as the DN specified in the `gsk_secure_soc_init()` initialization data.

**User Response:** Either remove the duplicate certificates or specify a label instead of a DN in the `gsk_secure_soc_init()` initialization data.

---

### 16 Incorrect key database password.

**Explanation:** The System SSL runtime is unable to decrypt a key database entry. Either the supplied database password is incorrect or the database is damaged.

**User Response:** Ensure that the correct key database password is used. Recreate the database if the error persists.

---

### 17 Key database password is expired.

**Explanation:** The key database password is expired.

**User Response:** Use the `gskkyman` command to assign a new password for the key database.

---

### 18 No certification authority certificates.

**Explanation:** The key database or SAF key ring does not contain any valid certification authority certificates. The SSL runtime needs at least one CA or self-signed certificate in order to perform client authentication.

**User Response:** Add the necessary certificates to the key database or SAF key ring and ensure that existing certificates are valid and have not expired.

---

### 19 No certificates available.

**Explanation:** The key database or SAF key ring does not contain any certificates.

**User Response:** Add the user certificate and any necessary certification authority certificates to the key database or SAF key ring.

---

### 70 Application is not APF-authorized.

**Explanation:** The `gsk_srb_initialize()` routine is called but the program is not APF authorized. SRB mode cannot be used by unauthorized applications.

**User Response:** Contact your system programmer to

get your application authorized.

---

**71 Unable to establish ESTAE environment.**

**Explanation:** The `gsk_srb_initialize()` routine is unable to establish the ESTAE error recovery environment.

**User Response:** Contact your service representative.

---

**72 Unable to create service thread.**

**Explanation:** The `gsk_srb_initialize()` routine is unable to create a thread to handle SRB processing.

**User Response:** Ensure that POSIX thread support is available to the application environment. Contact your service representative if the error persists.

---

**100 Initialization parameter is not valid**

**Explanation:** An initialization parameter for `gsk_initialize()` or `gsk_secure_soc_init()` is not valid.

**User Response:** Ensure that all of the parameters are correct. Contact your service representative if the error persists.

---

**102 Security type is not valid**

**Explanation:** The security type specified in the initialization data for the `gsk_initialize()` routine is not valid.

**User Response:** Specify a valid security type for the `sec_types` parameter.

---

**103 SSL V2 session timeout is not valid.**

**Explanation:** The SSL V2 session timeout specified in the initialization data for the `gsk_initialize()` routine is not valid.

**User Response:** Specify a valid SSL V2 session timeout value.

---

**104 SSL V3 session timeout is not valid.**

**Explanation:** The SSL V3 session timeout specified in the initialization data for the `gsk_initialize()` routine is not valid.

**User Response:** Specify a valid SSL V3 session timeout value.

---

**-1 No SSL cipher specifications.**

**Explanation:** The client and server cipher specifications do not contain at least one value in common. This error can also occur if no SSL protocols are enabled or if all of the enabled protocols have empty cipher specifications.

**User Response:** Ensure that the client and the server have at least one cipher specification in common.

---

**-2 No certificate received from partner.**

**Explanation:** The required certificate was not received from the communication partner.

**User Response:** Ensure that the remote application is sending the certificate. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**-3 Certificate key is not compatible with cipher suite.**

**Explanation:** The certificate key is not compatible with the negotiated cipher suite. The server certificate must have an RSA or DSA key while the client certificate may have an RSA or DSA key. This error can also occur if the client certificate has a DSA key but the server does not support DSA keys, the server key usage certificate extension does not allow key encipherment, or the client key usage certificate extension does not allow digital signature. For the 40-bit export ciphers, the server key usage certificate extension must allow digital signature.

**User Response:** Specify a certificate with the appropriate key type and key usage.

---

**-5 SSL V2 header is not valid.**

**Explanation:** The received message does not start with a valid SSL V2 header. This error can occur if an SSL V3 client attempts to establish a secure connection with an SSL V2 server.

**User Response:** Enable the SSL V2 protocol on the client and then retry the request.

---

**-6 Certificate format is not supported.**

**Explanation:** The certificate received from the communication partner is not supported by the current version of the System SSL runtime.

**User Response:** Collect a System SSL trace containing a dump containing the unsupported certificate and then contact your service representative.

---

**-7 Session renegotiation is not allowed.**

**Explanation:** An attempt to renegotiate the session parameters for an active connection is rejected by the peer application.

**User Response:** SSL processing continues using the current session parameters.

---

**-9 Certificate is revoked.**

**Explanation:** The certificate has been revoked by the certification authority.

**User Response:** Obtain a new certificate.

---

**-10 Error while reading or writing data.**

**Explanation:** An I/O error was reported while the System SSL runtime was reading or writing data.

**User Response:** Ensure that there are no network errors. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**-11 SSL message format is incorrect.**

**Explanation:** An incorrectly formatted SSL message is received from the communication partner.

**User Response:** Collect a System SSL trace containing a dump of the SSL message and then contact your service representative.

---

**-12 Message authentication code is incorrect.**

**Explanation:** The message authentication code (MAC) for a message is not correct. This indicates the message was modified during transmission.

**User Response:** Collect a System SSL trace containing a dump of the message and then contact your service representative if the error persists.

---

**-13 SSL protocol or certificate type is not supported.**

**Explanation:** The SSL handshake is not successful due to an unsupported protocol or certificate type. This error can occur if there is no enabled SSL protocol shared by both the client and the server.

**User Response:** Ensure that the desired SSL protocol is enabled on both the client and the server. Collect a System SSL trace containing a dump of the failing handshake and then contact your service representative if the problem persists.

---

**-14 Certificate signature is incorrect**

**Explanation:** The certificate signature is not correct for a certificate received from the communication partner.

**User Response:** Ensure that a valid certificate is being sent by the communication partner. Collect a System SSL trace containing a dump of the incorrect certificate and then contact your service representative if the error persists.

---

---

**-15 Certificate is not valid**

**Explanation:** Either the local certificate or the peer certificate is not valid.

**User Response:** Ensure that a valid certificate is being sent by the communication partner. Collect a System SSL trace containing a dump of the incorrect certificate and then contact your service representative if the error persists.

---

**-16 SSL protocol violation.**

**Explanation:** The communication partner has violated the SSL protocol by sending a message out of sequence or by omitting a required field from a message.

**User Response:** Collect a System SSL trace and then contact your service representative.

---

**-17 Permission denied.**

**Explanation:** The System SSL runtime is unable to access a file or system facility.

**User Response:** Ensure the application is authorized to access the file or facility. Collect a System SSL trace and then contact your service representative if the error persists.

---

**-18 Self-signed certificate cannot be validated.**

**Explanation:** A self-signed certificate cannot be validated because it is not in the key database or SAF key ring.

**User Response:** Add the self-signed certificate to the key database or SAF key ring.

---

**-19 Certification authority is unknown**

**Explanation:** The key database does not contain a certificate for the certification authority.

**User Response:** Obtain the certificate for the certification authority and add it to the key database.

---

**-20 Insufficient storage is available.**

**Explanation:** The System SSL runtime library is unable to obtain storage for an internal control block.

**User Response:** Increase the storage available to the application and then retry the failing operation.

---

**-21 Handle is in the incorrect state.**

**Explanation:** The SSL connection handle is in the incorrect state for the requested operation.

**User Response:** Correct the application to request SSL functions in the proper sequence.

---

---

**-22 Socket closed by remote partner.**

**Explanation:** The remote partner closed the socket.

**User Response:** None.

---

**-25 Certificate is expired or is not valid yet.**

**Explanation:** The current time is either before the certificate start time or after the certificate end time.

**User Response:** Obtain a new certificate if the certificate is expired or wait until the certificate becomes valid if it is not valid yet.

---

**-26 Key exceeds allowable export size.**

**Explanation:** The key size used for an export cipher suite exceeds the allowable maximum size. For RSA and DSA keys, the maximum export key size is 512 bits. If the certificate key is larger than 512 bits, the SSL runtime will use a temporary 512-bit key for the connection.

**User Response:** Collect a System SSL trace and then contact your service representative.

---

**-27 Key entry does not contain a private key.**

**Explanation:** The key entry does not contain a private key. This error can also occur if the private key is stored in ICSF and ICSF services are not available.

**User Response:** Specify a key entry containing a private key value. Ensure that the ICSF started task is running if the private key is stored in ICSF.

---

**-28 Function parameter is not valid.**

**Explanation:** A parameter specified on an SSL function call is not valid.

**User Response:** Ensure that the parameters on the failing function call are correct. Contact your service representative if the error persists.

---

**-30 Socket request would block.**

**Explanation:** The socket is in non-blocking mode and the socket request returned the EWOULDBLOCK error.

**User Response:** Retry the `gsk_secure_soc_read()` or `gsk_secure_soc_write()` request when the socket is ready to send or receive data.

---

**-34 Certificate revocation list cannot be processed.**

**Explanation:** A certificate revocation list (CRL) is not valid and cannot be processed.

**User Response:** Contact the certification authority and obtain a replacement CRL.

---

---

**-35 Certificate validation error.**

**Explanation:** An error is detected while validating a certificate. This error can occur if a root CA certificate is not found in the key database or SAF keyring or if the certificate is not marked as a trusted certificate.

**User Response:** Verify that the root CA certificate is in the key database or SAF keyring and is marked as trusted. Check all certificates in the certification chain and verify that they are trusted and are not expired. Collect a System SSL trace containing the error and then contact your service representative if the problem persists.

---

**-36 Cryptographic processing error.**

**Explanation:** An error is detected by a cryptographic function.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**-37 ASN processing error.**

**Explanation:** An error is detected while processing a certificate field.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**-38 LDAP processing error.**

**Explanation:** An error is detected while setting up the LDAP environment or retrieving an LDAP directory entry.

**User Response:** Ensure that the LDAP server is running and that there are no network errors. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**-29 LDAP is not available.**

**Explanation:** The System SSL runtime is unable to access the LDAP server.

**User Response:** Ensure that the LDAP server is running and that there are no network problems. Collect a System SSL trace and then contact your service representative if the error persists.

---

**-40 SSL V2 cipher is not valid.**

**Explanation:** The SSL V2 cipher is not valid.

**User Response:** Specify a valid cipher.

---

---

**-41 SSL V3 cipher is not valid.**  
**Explanation:** The SSL V3 cipher is not valid.  
**User Response:** Specify a valid cipher.

---

**-42 Bad handshake specification.**  
**Explanation:** The handshake specification for the `gsk_secure_soc_init()` routine is not valid.  
**User Response:** Specify a valid value for the `hs_type` field in the `gsk_secure_soc_init()` initialization data.

---

**-43 No read function.**  
**Explanation:** No read function is provided for the `gsk_secure_soc_init()` routine.  
**User Response:** Specify a read function for the `skread` field in the `gsk_secure_soc_init()` initialization data.

---

**-44 No write function.**  
**Explanation:** No write function is provided for the `gsk_secure_soc_init()` routine.  
**User Response:** Specify a write function for the `skwrite` field in the `gsk_secure_soc_init()` initialization data.

---

**-46 Socket write request would block.**  
**Explanation:** A socket write request issued as part of an SSL handshake return the EWOULDBLOCK error.  
**User Response:** Retry the failing request when the socket is ready to send data.

---

**-47 Connection is active.**  
**Explanation:** An SSL secure connection operation cannot be completed due to an active request for the connection.  
**User Response:** Retry the failing request when the currently active request has completed.

---

**-48 Connection closed.**  
**Explanation:** For `gsk_secure_soc_read()`, a close notification has been received from the peer application. For `gsk_secure_soc_write()`, a close notification has been sent to the peer application. A close notification is sent when a close notification is received from the peer application. Additional data may not be sent by the application after the close notification has been sent to the peer application.  
**User Response:** None.

---

**-51 Protocol is not SSL V3 or TLS V1.**  
**Explanation:** The requested function requires either the SSL V3 or the TLS V1 protocol.  
**User Response:** Ensure that the correct protocol is in use before issuing the request.

---

**-53 Internal error reported by remote partner.**  
**Explanation:** The peer application has detected an internal error while performing an SSL operation and has sent an alert to close the secure connection.  
**User Response:** Check the error log for the remote application to determine the nature of the processing error.

---

**-54 Unknown alert received from remote partner.**  
**Explanation:** The peer application has sent an alert message which is not recognized by the System SSL runtime.  
**User Response:** Collect a System SSL trace and then contact your service representative.

---

**-70 SRB processing is not initialized.**  
**Explanation:** The `gsk_srb_initialize()` routine has not been called to initialize the SRB support.  
**User Response:** Call `gsk_srb_initialize()` before making any calls to GSKSRBRD or GSKSRBWT.

---

**-71 SRB lock timeout.**  
**Explanation:** The GSKSRBRD or GSKSRBWT routine is unable to obtain the lock for the SRB control area.  
**User Response:** Ensure that the SRB processing threads are not suspended (for example, a synchronous dump will suspend thread execution while the dump is processed). Contact your service representative if the error persists.

---

**-72 SRB suspend failed.**  
**Explanation:** The GSKSRBRD or GSKSRBWT routine is unable to suspend execution while waiting for the completion of the read or write request.  
**User Response:** Contact your service representative.

---

**-73 Unknown SRB service request.**  
**Explanation:** The SRB service task does not recognized the function request.  
**User Response:** Contact your service representative.

---

---

**-99**            **An unexpected error has occurred.**

**Explanation:** An unexpected error is detected by the System SSL runtime.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

**-100**           **Buffer size is not valid.**

**Explanation:** The socket buffer or buffer size is not valid.

**User Response:** Specify a valid buffer and buffer size.

---

**-101**           **Handle is not valid.**

**Explanation:** The SSL connection handle specified on a System SSL function call is not valid.

**User Response:** Call the `gsk_secure_soc_init()` function to create an SSL connection handle.

---

## ASN.1 Status Codes (014CExxx)

This section describes the ASN.1 status codes.

---

### 014CE001 No more data.

**Explanation:** The end of an ASN.1 encoded stream is reached prematurely. This error can occur if an encoded stream is truncated.

**User Response:** Verify that the encoded certificate is not modified. Contact your service representative if the error persists.

---

### 014CE002 Data value overflow.

**Explanation:** A decoded data value is too large to be represented as the specified data type.

**User Response:** Contact your service representative.

---

### 014CE003 Length value is not valid.

**Explanation:** The length of an encoded item is not valid. This error can occur if an encoded stream is truncated.

**User Response:** Verify that the encoded certificate is not modified. Contact your service representative if the error persists.

---

### 014CE004 Data encoding is not valid.

**Explanation:** The encoded data violates the ASN.1 encoding rules.

**User Response:** Contact your service representative.

---

### 014CE005 Parameter is not valid

**Explanation:** An application parameter is not valid.

**User Response:** Correct the application to specify valid parameters for the failing function call. Contact your service representative if the error persists.

---

### 014CE006 Insufficient memory is available.

**Explanation:** There is not enough memory available to allocate a required control block or data element.

**User Response:** Increase the memory available to the application and then retry the request. Contact your service representative if the error persists.

---

### 014CE007 Indefinite-length encoding is not allowed

**Explanation:** An indefinite-length encoding is encountered for a data element that requires a length value.

**User Response:** Contact your service representative.

---

### 014CE008 Data element must be an ASN.1 primitive.

**Explanation:** A constructed element is encountered instead of an ASN.1 primitive.

**User Response:** Contact your service representative.

---

### 014CE009 Data element must be constructed.

**Explanation:** An ASN.1 primitive is encountered instead of a constructed element.

**User Response:** Contact your service representative.

---

### 014CE00A Data value is not present

**Explanation:** An ASN.1 element has no value and does not have a default value.

**User Response:** Contact your service representative.

---

### 014CE00B Indefinite-length encoding is not supported.

**Explanation:** Indefinite-length encoding is not supported for the current structure. An X.509 certificate is encoded using ASN.1 DER (Distinguished Encoding Rules) which does not allow the use of indefinite-length encodings.

**User Response:** Contact your service representative.

---

### 014CE00C Unused bit count is not valid

**Explanation:** The unused bit count for a bit string must be between 0 and 7.

**User Response:** Contact your service representative if this error occurs while decoding a bit string. Correct the application if this error occurs while encoding a bit string.

---

### 014CE00D Unused bit count is not valid for a segmented bit string.

**Explanation:** The unused bit count must be zero for each segment other than the final segment of a bit string.

**User Response:** Contact your service representative.

---

### 014CE00E Data type is not correct.

**Explanation:** An unexpected data type is encountered while decoding a data element.

**User Response:** Contact your service representative.

---

**014CE00F Excess data found at end of data element**

**Explanation:** There is unprocessed encoded data after decoding a data element.

**User Response:** Contact your service representative.

---

**014CE010 Required data element is missing.**

**Explanation:** A required data element is not found when decoding an encoded structure.

**User Response:** Contact your service representative.

---

**014CE011 Selection is not within the valid range.**

**Explanation:** The selection for an ASN.1 element is not within the valid range for that element.

**User Response:** Contact your service representative.

---

**014CE012 No selection found**

**Explanation:** No selection found for an ASN.1 element.

**User Response:** Contact your service representative.

---

**014CE013 Syntax already set.**

**Explanation:** The decoding syntax has already been set for an ASN.1 element.

**User Response:** Contact your service representative.

---

**014CE014 Character string cannot be converted.**

**Explanation:** A character string cannot be converted to the target code page. This error can occur when a character string contains characters which cannot be represented in the target code page.

**User Response:** Ensure that the character string uses characters which are valid for the target code page. Contact your service representative if the error persists.

---

**014CE015 Codeset is not allowed**

**Explanation:** The requested codeset is not valid for the current data element.

**User Response:** Contact your service representative.

---

**014CE016 Attribute value is not valid.**

**Explanation:** An attribute value is not valid.

**User Response:** Contact your service representative.

---

---

**014CE017 Attribute value separator is missing.**

**Explanation:** An X.500 attribute value separator is missing.

**User Response:** Ensure that the name string is correctly formed. Each attribute consists of an attribute type and an attribute value separated by an equal sign. Contact your service representative if the error persists.

---

**014CE018 Attribute value is missing**

**Explanation:** An X.500 attribute value is missing.

**User Response:** Correct the application to specify an attribute for each relative distinguished name.

---

**014CE019 Object identifier syntax error**

**Explanation:** The syntax of an object identifier is not valid. The object identifier consists of one or more decimal numbers separated by periods.

**User Response:** Correct the application to specify a valid object identifier.

---

**014CE01A PKCS12 version is not correct.**

**Explanation:** The PKCS12 version is not correct.

**User Response:** Contact your service representative.

---

**014CE01B Interval is not valid.**

**Explanation:** The certificate interval is not valid.

**User Response:** Contact your service representative.

---

**014CE01C Object identifier element count is not valid**

**Explanation:** An object identifier must have at least three elements.

**User Response:** Correct the application to provide a valid object identifier.

---

**014CE01D Incorrect value for the first object identifier element.**

**Explanation:** The first element of an object identifier must be 0, 1, or 2.

**User Response:** Correct the application to provide a valid object identifier.

---

**014CE01E Incorrect value for the second object identifier element.**

**Explanation:** The second element of an object identifier must be between 0 and 39 if the first element is 0 or 1.

---

**User Response:** Correct the application to provide a valid object identifier.

---

**014CE01F Unknown attribute identifier.**

**Explanation:** An unrecognized attribute identifier is encountered while decoding a certificate extension or an X.509 name. As a result, the attribute value cannot be decoded.

**User Response:** Ensure that the name string is correctly formed. Each attribute consists of an attribute type and an attribute value separated by an equal sign. Contact your service representative if the error persists.

---

**014CE020 Unknown critical certificate extension.**

**Explanation:** The X.509 certificate contains a critical extension that is not recognized by the System SSL runtime. The certificate cannot be processed.

**User Response:** Obtain a new certificate without the unknown critical certificate extension.

---

**014CE021 X.500 name syntax error.**

**Explanation:** The syntax of an X.500 distinguished name is not valid. Refer to RFC 2253 (String Representation of Distinguished Names) for more information on the format of a distinguished name.

**User Response:** Correct the application to specify a valid distinguished name.

---

## CMS Status Codes (03353xxx)

This section describes some CMS status codes.

---

### 03353001 Insufficient memory is available.

**Explanation:** There is not enough memory available to allocate a required control block or data element.

**User Response:** Increase the memory available to the application and then retry the request. Contact your service representative if the error persists.

---

### 03353002 Certificate extension is not supported.

**Explanation:** An X.509 certificate extension is either not supported by the current level of the System SSL runtime or is not supported by the certificate version. The certificate extension is not processed. If the extension is marked as a critical extension, the X.509 certificate cannot be processed.

**User Response:** Upgrade the System SSL runtime if a later software level supports the certificate extension.

---

### 03353003 Cryptographic algorithm is not supported.

**Explanation:** An X.509 cryptographic algorithm is not supported by the current level of the System SSL runtime. This error can also occur if the current operation does not support the specified cryptographic algorithm.

**User Response:** Ensure that the cryptographic algorithm is supported for the requested operation. Upgrade the System SSL runtime if a later software level supports the cryptographic algorithm.

---

### 03353004 Signature is not correct

**Explanation:** The signature is incorrect for an X.509 certificate or certificate revocation list. This usually means the certificate has been modified since it was signed by the issuing Certificate Authority.

**User Response:** Verify that the certificate has not been modified. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

### 03353005 Cryptographic request failed.

**Explanation:** A cryptographic request failed with an unexpected error. This error can occur if the hardware cryptographic support becomes unavailable after the application has been initialized.

**User Response:** Collect a System SSL trace containing the error and then contact your service representative.

---

### 03353006 Input/Output request canceled.

**Explanation:** An input/output operation is canceled by the user. This can occur if the user cancels a terminal read request by pressing an attention key or by pressing the enter key without entering any data.

**User Response:** None

---

### 03353007 Input/Output request failed.

**Explanation:** An input/output operation fails.

**User Response:** Verify that the file or keyring can be accessed and is not damaged. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

### 03353008 Verification password does not match.

**Explanation:** The user is prompted to verify the password by entering it a second time. The user did not enter the same password both times.

**User Response:** Enter the same password when prompted.

---

### 03353009 File or keyring not found

**Explanation:** A file or keyring cannot be opened because it is not found.

**User Response:** Verify that the correct name is specified. Contact your service representative if the error persists.

---

### 0335300A Database is not valid.

**Explanation:** The key database or the request database is not valid. This error can occur if the wrong database password is used when opening the database or if the database format is not supported by the current level of the System SSL runtime.

**User Response:** Verify that the database has not been modified or truncated. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

### 0335300B Message not found.

**Explanation:** The System SSL runtime is unable to locate a message in the message catalog.

**User Response:** Verify that the message catalog can be accessed by the application and can be located using the NLSPATH environment variable. Contact your service representative if the error persists.

---

**0335300C Handle is not valid.**

**Explanation:** The handle passed to the System SSL runtime is not valid. This error can occur if the handle has been closed or is not the proper type for the requested function.

**User Response:** Pass a valid handle to the System SSL routine.

---

**0335300D Record deleted.**

**Explanation:** The requested record is deleted.

**User Response:** None

---

**0335300E Record not found.**

**Explanation:** The requested record is not found.

**User Response:** None

---

**0335300F Incorrect database type**

**Explanation:** The database does not support the requested operation. This error can occur if the database type is not valid. It can also occur if an attempt is made to add a request record to a key database or a key record to a request database.

**User Response:** Specify an operation supported by the database.

---

**03353010 Database is not open for update.**

**Explanation:** A request to modify the key or request database cannot be completed because update mode was not requested when the database was opened.

**User Response:** Request update mode when opening a database for modification.

---

**03353011 Mutex request failed.**

**Explanation:** A mutex operation failed.

**User Response:** Contact your service representative.

---

**Backup file already exists.**

**Explanation:** Before updating a database file, the System SSL runtime creates a backup file with the same name with ".new" appended to the name. This file is then deleted after the database file has been rewritten. The file is not deleted if an error occurs while rewriting the database file.

**User Response:** Correct the problem that caused the database update to fail. Then copy the backup file to the database file and delete the backup file.

---

---

**03353013 Database already exists.**

**Explanation:** A request to create a new database cannot be completed because the database file already exists.

**User Response:** Choose a different name for the new database or delete the existing database.

---

**03353014 Record is too big.**

**Explanation:** A new record cannot be added to the database because it is larger than the database record length.

**User Response:** Create a new database with a larger record length.

---

**03353015 Database password is expired.**

**Explanation:** The database password is expired.

**User Response:** Change the database password.

---

**03353016 The password is not correct.**

**Explanation:** The wrong password is specified for a key database, an encrypted private key, or an import file. This error can also occur if the file has been modified.

**User Response:** Specify the correct password.

---

**03353017 Access denied.**

**Explanation:** The database or keyring cannot be opened because the permissions do not allow access by the current user.

**User Response:** Ensure that the user has read/write access to the database if opening the database for update mode; otherwise ensure that the user has read access to the database or keyring.

---

**03353018 Database is locked for update.**

**Explanation:** Another process has opened the database in update mode. Only one process may have the database open for update at a time.

**User Response:** Wait until the database has been closed by the other process and then retry the request.

---

**03353019 Record length is too small.**

**Explanation:** The database record length is less than the minimum value of 2500.

**User Response:** Specify a record length of 2500 or greater.

---

---

**0335301A No private key.**

**Explanation:** A private key request cannot be processed because the database entry does not contain a private key. This error can occur if the private key is stored in the Integrated Cryptographic Service Facility (ICSF) but the CSF started task is not running.

**User Response:** Verify that the CSF started task is running if the private key is stored in ICSF. Otherwise, repeat the failing request using a database entry containing a private key.

---

**0335301B Record label is not valid.**

**Explanation:** The record label is not valid. A label may contain letters, numbers, and punctuations. A record label may not be an empty string.

**User Response:** Provide a valid record label.

---

**0335301C Record label is not unique..**

**Explanation:** A record label must be unique with a database.

**User Response:** Provide a unique record label.

---

**0335301D Record type is not valid.**

**Explanation:** The database record type is not valid.

**User Response:** Provide a valid database record type.

---

**0335301E Duplicate certificate.**

**Explanation:** An attempt is made to add a certificate to a key database but the database already contains the certificate. A certificate is a duplicate if the issuer name and certificate serial number are the same.

**User Response:** Delete the existing certificate before adding the new certificate.

---

**0335301F Incorrect Base64 encoding.**

**Explanation:** An encoded stream cannot be decoded because it contains an incorrect Base64 encoding. A Base64 encoding consists of a header line, encoded text, and a footer line. The encoded text is encoded using a 64-character subset in groups of 4 characters.

**User Response:** Ensure that the encoded stream has not been truncated or modified. Base64 encoding uses text data and must be in the local code page. Contact your service representative if the error persists.

---

**03353020 Unrecognized file or message encoding.**

**Explanation:** A file or message cannot be imported because the format is not recognized.

System SSL supports X.509 DER-encoded certificates,

PKCS #7 signed data messages, and PKCS #12 personal information exchange messages for certificate import files. The import file data may be the binary data or the Base64-encoding of the binary data.

System SSL supports PKCS #7 data, encrypted data, signed data, and enveloped data for messages. This error can also occur if the message is not constructed properly.

**User Response:** Ensure that the import file or message has not been modified. A Base64-encoded import file must be converted to the local code page when it is moved to another system while a binary import file must not be modified when it is moved to another system.

---

**03353021 Certificate is not yet valid.**

**Explanation:** The current time is earlier than the beginning of the certificate validity.

**User Response:** Either wait until the certificate is valid or request a new certificate with an earlier starting date from the certification authority.

---

**03353022 Certificate is expired**

**Explanation:** The current time is after the end of the certificate validity.

**User Response:** Request a new certificate from the certification authority.

---

**03353023 Name format is not supported.**

**Explanation:** An unsupported name format is encountered while validating a certificate.

**User Response:** Contact your service representative.

---

**03353024 Issuer certificate not found.**

**Explanation:** An issuer certificate is not found while validating a certificate. This error can occur if the key database contains the issuer certificate but the certificate is not trusted or has expired.

**User Response:** Ensure that the key database contains a certificate for the issuer of the certificate being validated and that the certificate is marked as trusted. Contact your service representative if the error persists.

---

**03353025 Certification path is too long.**

**Explanation:** The certification path length exceeds the maximum specified in the certification authority certificate.

**User Response:** Report the problem to the certification authority.

---

---

**03353026 Incorrect key usage.**

**Explanation:** The key usage certificate extension does not permit the requested key operation.

**User Response:** Obtain a certificate which allows the desired key operation.

---

**03353027 Issuer is not a certification authority.**

**Explanation:** The issuer of an X.509 certificate is not a certification authority. This indicates that the basic constraints certificate extension in the issuer certificate does not contain the certification authority indicator.

**User Response:** Report the problem to the issuer of the certificate.

---

**03353028 Export file format is not supported.**

**Explanation:** The requested export file format is not supported for the specified database record. Certificates can be exported using the DER and PKCS #7 formats. Certificates and keys can be exported using the PKCS #12 formats.

**User Response:** Select an appropriate export file format.

---

**03353029 Cryptographic algorithm is not available.**

**Explanation:** An X.509 cryptographic algorithm is not available. Due to government export regulations, strong encryption is not available on the local system.

**User Response:** Select an algorithm that is available.

---

**0335302A Record type cannot be changed.**

**Explanation:** The record type cannot be changed when replacing a database record.

**User Response:** Create a new database entry for the record.

---

**0335302B Subject name cannot be changed.**

**Explanation:** The subject name cannot be changed when replacing a database record.

**User Response:** Create a new database entry for the record.

---

**0335302C Public key cannot be changed.**

**Explanation:** The subject public key cannot be changed when replacing a database record.

**User Response:** Create a new database entry for the record.

---

---

**0335302D Default key cannot be changed**

**Explanation:** The default key for the database cannot be changed using the `gsk_replace_record()` routine.

**User Response:** Use the `gsk_set_default_key()` routine to change the default key for the database.

---

**0335302E Database contains certificates signed by the certificate.**

**Explanation:** A CA certificate cannot be deleted because the database still contains certificates that were signed using the certificate.

**User Response:** Delete all certificates signed by the CA certificate before deleting the certificate.

---

**0335302F Certificate chain is not trusted.**

**Explanation:** A certification authority (CA) certificate in the certification chain is not trusted.

**User Response:** Set the trust status for the CA certificate if the certificate can be used for authentication purposes.

---

**03353030 Key not supported by encryption or signature algorithm.**

**Explanation:** The supplied key is not supported by the requested encryption or signature algorithm. For example, an RSA key cannot be used to verify a DSA signature and a DSA key cannot be used to encrypt data.

**User Response:** Provide the appropriate key for the encryption or signature algorithm.

---

**03353031 Signer certificate not found.**

**Explanation:** A signer certificate is not found while creating or processing a signed message,

**User Response:** Provide a certificate for each signer.

---

**03353032 Content type is not supported.**

**Explanation:** An unsupported PKCS #7 content type is encountered.

**User Response:** Refer to the Programming Reference for the failing routine to determine the supported content types.

---

**03353033 Recipient certificate not found.**

**Explanation:** A recipient certificate is not found while creating or processing an enveloped message.

**User Response:** Provide at least one recipient certificate.

---

---

**03353034 Encryption key size is not supported.**

**Explanation:** The encryption key size is not supported by the System SSL runtime.

**User Response:** Refer to the System SSL documentation to determine which key sizes are supported. In general, 40-bit keys and 128-bit keys are supported for RC2 and RC4, 56-bit keys are supported for DES, and 168-bit keys are supported for Triple DES. RSA keys must be between 512 and 2048 bits while DSA keys must be between 512 and 1024 bits.

---

**03353035 Encryption key parity is not correct.**

**Explanation:** DES and Triple DES encryption keys must have odd parity for each key byte.

**User Response:** Verify that the key is generated correctly. Contact your service representative if the error persists.

---

**03353036 Encryption key is weak.**

**Explanation:** A small subset of the possible DES and Triple DES encryption keys are weak and can be broken more easily than the rest of the keys. For this reason, the weak keys should be avoided when generating a DES or Triple DES key.

**User Response:** Contact your service representative.

---

**03353037 Initial vector size is not correct.**

**Explanation:** The initial vector used by the encryption routine is not the correct length.

**User Response:** Contact your service representative.

---

**03353038 Encryption data size is not correct.**

**Explanation:** The length of the encryption data is not correct. For symmetric key algorithms using cipher block chaining, the encryption data must be a multiple of the cipher block size. For asymmetric key algorithms, the encryption data must be the same length as the cipher key modulus.

**User Response:** Verify that the encryption data has not been truncated. Contact your service representative if the error persists.

---

**03353039 Encryption block format is not correct.**

**Explanation:** The encryption block format is not correct following decryption. This error can occur if the wrong key is used to decrypt the block.

**User Response:** Verify that the correct key is being used to decrypt the data. Contact your service representative if the error persists.

---

**0335303A Number does not have a modular inverse.**

**Explanation:** The cryptographic support is unable to find an inverse for a number.

**User Response:** Contact your support representative.

---

**0335303B LDAP processing error.**

**Explanation:** An error is detected while setting up the LDAP environment or retrieving an LDAP directory entry.

**User Response:** Ensure that the LDAP server is running and that there are no network errors. Collect a System SSL trace containing the error and then contact your service representative if the error persists.

---

**0335303C LDAP is not available.**

**Explanation:** The System SSL runtime is unable to access the LDAP server.

**User Response:** Ensure that the LDAP server is running and that there are no network problems. Collect a System SSL trace and then contact your service representative if the error persists.

---

**0335303D Digest data size is not correct.**

**Explanation:** The length of the digest data is not correct. The digest data size is 16 bytes for the MD2 and MD5 algorithms and 20 bytes for the SHA-1 algorithm.

**User Response:** Verify that the data has not been truncated. Contact your service representative if the error persists.

---

**0335303E Database name is not valid.**

**Explanation:** The database file name or SAF key ring name is not valid. The length of the fully-qualified database file name must be between 1 and 251 while the length of the SAF key ring must be between 1 and 237.

**User Response:** Provide a valid database name.

---

**0335303F Database open failed.**

**Explanation:** The System SSL runtime is unable to open the HFS database file or the SAF key ring.

**User Response:** Verify that the database file or SAF key ring exists and is accessible by the application. Collect a System SSL trace and then contact your service representative if the error persists.

---

**03353040 Self-signed certificate not in database.**

**Explanation:** A self-signed certificate cannot be validated because it is not in the key database or SAF key ring.

**User Response:** Add the self-signed certificate to the key database or SAF key ring.

---

**03353041 Certificate is revoked.**

**Explanation:** A certificate is revoked and cannot be used.

**User Response:** Obtain a new certificate from the certification authority.

---

**03353042 Issuer name is not valid.**

**Explanation:** The certificate issuer name must be a non-empty X.509 distinguished name.

**User Response:** Obtain a new certificate with a valid issuer name.

---

**03553043 Subject name is not valid.**

**Explanation:** The certificate subject name must be either a non-empty distinguished name or an empty distinguished name with a SubjectAltName certificate extension.

**User Response:** Obtain a new certificate with a valid subject name.

---

**03353044 Name constraints violated.**

**Explanation:** The certificate name is not allowed by the certification path name constraints.

**User Response:** Report the problem to the certification authority.

---

**03353045 No content data.**

**Explanation:** The PKCS #7 content information does not contain any content data.

**User Response:** Change the application to provide content data for the content information.

---

**03353046 Message version is not supported.**

**Explanation:** An unsupported message version is encountered.

**User Response:** Refer to the Programming Reference for the failing routine to determine the supported message versions.

---

---

**03353047 Subject name is same as signer name.**

**Explanation:** A request to create a new certificate cannot be processed because the requested subject name is the same as the subject name of the signing certificate.

**User Response:** Choose a different subject name for the new certificate.

---

## SSL Started Task Messages (GSK01nnn)

This section describes SSL started task messages.

---

### **GSK01001I System SSL version** *version.release* **Service level** *level*.

**Explanation:** This message displays the System SSL version, release, and service level.

**User Response:** None

---

### **GSK01002E Insufficient storage available.**

**Explanation:** The SSL server is unable to obtain storage for an internal control block.

**User Response:** Increase the storage available to the GSKSRVR started task and then retry the request.

---

### **GSK01003I SSL server initialization complete.**

**Explanation:** The server initialization is complete.

**User Response:** None

---

### **GSK01004I SSL server shutdown requested.**

**Explanation:** The system operator has entered a STOP command for the SSL server.

**User Response:** None

---

### **GSK01005E Unrecognized SSL server command:** **Specify DISPLAY, TRACE or STOP.**

**Explanation:** An unrecognized command name is specified on a MODIFY operator command. The valid SSL server commands are DISPLAY, TRACE and STOP.

**User Response:** Specify a valid SSL server command.

---

### **GSK01006E Incorrect command option specified.**

**Explanation:** An incorrect SSL server command option is specified.

The valid DISPLAY command options are:

- CRYPTO - Display the available encryption algorithms.
- LEVEL - Display the System SSL version, release, and service level.
- SIDCACHE - Display the sysplex session cache status.
- XCF - Display SSL sysplex status.

The valid TRACE command options are:

- OFF - Turn off SSL tracing
- ON,level - Enable SSL tracing using the specified trace level.

**User Response:** Specify a valid command option.

---

### **GSK01007E Missing command option.**

**Explanation:** An SSL server command is entered which requires a command option but no command option is entered.

**User Response:** Enter a complete SSL server command.

---

### **GSK01008I Sysplex status.**

**Explanation:** This message is displayed in response to the SSL server DISPLAY XCF command. The remaining lines in this multi-line message display the status of each SSL server in the sysplex. A server is ACTIVE if the GSKSRVR started task is running. A security server is INACTIVE if the GSKSRVR started task has been stopped. No entry is displayed for a system where the GSKSRVR started task has not been started.

**User Response:** None

---

### **GSK01009I Cryptographic status.**

**Explanation:** This message is displayed in response to the SSL server DISPLAY CRYPTO command. The remaining lines in this multi-line message display the available encryption algorithms.

**User Response:** None

---

### **GSK01010A The SSL server is already running.**

**Explanation:** The GSKSRVR started task is already running. Only one instance of the SSL server may be active in the same system.

**User Response:** Stop the GSKSRVR started task before starting a new instance of the SSL server.

---

### **GSK01011A The SSL server is not APF-authorized.**

**Explanation:** The GSKSRVR started task is not running with APF authorization.

**User Response:** Add the GSK.SGSKLOAD dataset to the list of APF-authorized datasets and then restart the GSKSRVR started task. If you are using a STEPLIB or JOBLIB for the GSKSRVR started task, verify that all datasets in the concatenation are APF-authorized.

---

### **GSK01012A Unable to make address space non-swappable: Error** *error-code*.

**Explanation:** The SSL server is unable to make its address space non-swappable. The error code is the value returned by the SYSEVENT system service.

**User Response:** Verify that the GSKSRVR started task is APF-authorized. Refer to the SYSEVENT description in *z/OS MVS Authorized Assembler Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01013I SSL server restart registration complete on system.**

**Explanation:** The GSKSRVR started task has successfully registered with ARM (Automatic Restart Management) on the indicated system. The GSKSRVR started task will be automatically restarted if it fails unexpectedly (it will not be restarted if it detects an error and stops). The ARM element type is SYSSSL and the ARM element name is GSKSRVR\_system-name. The ARM policy can be used to override the default registration values if needed.

**User Response:** None

---

**GSK01014I SSL server restarting on system.**

**Explanation:** The GSKSRVR started task is being restarted following an unexpected failure. The RESTART\_ATTEMPTS value in the ARM policy determines the number of restarts which will be attempted.

**User Response:** None

---

**GSK01015E Unable to register for restart: Error error-code, Reason reason-code.**

**Explanation:** The GSKSRVR started task is unable to register with ARM (Automatic Restart Management). The IXCARM request failed with the indicated error and reason codes.

**User Response:** Refer to the IXCARM description in *z/OS MVS Sysplex Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01016E Unable to deregister for restart: Error error-code, Reason reason-code.**

**Explanation:** The GSKSRVR started task is unable to deregister with ARM (Automatic Restart Management). The IXCARM request failed with the indicated error and reason codes.

**User Response:** Refer to the IXCARM description in *z/OS MVS Sysplex Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01017I SSL server restart deregistration complete on system.**

**Explanation:** The GSKSRVR started task has successfully deregistered with ARM (Automatic Restart

Management) on the indicated system. The SSL server will no longer be automatically restarted if it fails unexpectedly.

**User Response:** None

---

**GSK01018I Trace option processed: trace-option.**

**Explanation:** The indicated trace request has been processed by the SSL server.

**User Response:** None

---

**GSK01019E Unable to create mutex: error-text.**

**Explanation:** The GSKSRVR started task is unable to create a mutex for the indicated reason.

**User Response:** Refer to the pthread\_mutex\_init description in *z/OS C/C++ Run-Time Library Reference* for more information. Contact your service representative if the error persists.

---

**GSK01020E Unable to lock mutex: error-text.**

**Explanation:** The GSKSRVR started task is unable to lock a mutex for the indicated reason.

**User Response:** Refer to the pthread\_mutex\_lock description in *z/OS C/C++ Run-Time Library Reference* for more information. Contact your service representative if the error persists.

---

**GSK01021E Unable to create thread: error-text.**

**Explanation:** The GSKSRVR started task is unable to create a thread for the indicated reason.

**User Response:** Refer to the pthread\_create description in *z/OS C/C++ Run-Time Library Reference* for more information. Contact your service representative if the error persists.

---

**GSK01022E Unable to initialize local services: Error error-code, Reason reason-code.**

**Explanation:** The GSKSRVR started task is unable to initialize the local services support. The error code indicates the failing system function and the reason code is the error code returned by the system function.

The following error codes are defined:

- 1 = The job step is not APF-authorized.
- 2 = The security server is already running.
- 3 = The ESTAEX request failed.
- 5 = The LXRES request failed.
- 6 = The ETCRE request failed.
- 7 = The ETCON request failed.
- 8 = The IEANTCR request failed.
- 9 = The CTRACE DEFINE request failed.

**User Response:** Verify that the GSKSRVR started task is APF-authorized. Refer to the system function description in *z/OS MVS Authorized Assembler Services*

Reference for more information. Contact your service representative if the error persists.

---

**GSK01023E Unable to create session cache data space: Error *error-code*, Reason *reason-code*.**

**Explanation:** The GSKSRVR started task is unable to create the session cache data space.

The following error codes are defined:

**1 = DSPSERV CREATE failed.**

The reason code contains the DSPSERV return code in the upper halfword and bits 8-23 of the DSPSERV reason code in the lower halfword.

**2 = ALESERV ADD failed.**

The reason code is the ALESERV return code.

**User Response:** Refer to the DSPSERV or ALESERV description in *z/OS MVS Authorized Assembler Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01024E Unable to initialize cross-system services: Error *error-code*, Reason *reason-code*.**

**Explanation:** The GSKSRVR started task is unable to initialize cross-system services.

The following error codes are defined:

**1 = The job step is not APF-authorized.**

**3 = IXCJOIN failed.**

The reason code contains the IXCJOIN return code in the upper halfword and the IXCJOIN reason code in the lower halfword.

**4 = IXCQUERY failed.**

The reason code contains the IXCQUERY return code in the upper halfword and the IXCQUERY reason code in the lower halfword.

**User Response:** Refer to the IXCJOIN or IXCQUERY description in *z/OS MVS Sysplex Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01025I System *name* has joined the GSKSRVR group.**

**Explanation:** The GSKSRVR started task has completed initialization on the indicated system and is now a member of the GSKSRVGP cross-system group.

**User Response:** None

---

**GSK01026I System *name* has left the GSKSRVR group.**

**Explanation:** The GSKSRVR started task is stopping on the indicated system and has left the GSKSRVGP cross-system group.

**User Response:** None

---

**GSK01027I Cross-system services ended due to sysplex partitioning.**

**Explanation:** The local system is leaving the sysplex. As a result, GSKSRVR cross-system services are no longer available.

**User Response:** None

---

**GSK01028E Local program call request failed: Error *error-code*.**

**Explanation:** The GSKSRVR started task is unable to process a local program call request.

The following error codes are defined:

- 8 = Parameter buffer overflow.
- 12 = Unable to allocate storage.
- 16 = Local service support is not enabled.
- 20 = Program call task abended.
- 24 = Unable to obtain control lock.
- 28 = Requested function is not supported.

**User Response:** Contact your service representative.

---

**GSK01029I Cross-system services are not available.**

**Explanation:** The DISPLAY XCF command cannot be processed because cross-system services are not available.

**User Response:** None

---

**GSK01030I Maximum number of lines displayed.**

**Explanation:** The maximum number of lines allowed for a multi-line write-to-operator message has been reached.

**User Response:** None

---

**GSK01031I No session cache users.**

**Explanation:** The DISPLAY SIDCACHE command was issued but there are no session cache users to display.

**User Response:** None

---

**GSK01032I Session cache status**

**Explanation:** This message is displayed in response to the SSL server DISPLAY SIDCACHE command. The remaining lines in this multi-line message display the cache users.

**User Response:** None

---

**GSK01033E Unable to extend the session cache data space: Error *error-code*, Reason *reason-code*.**

**Explanation:** The GSKSRVR started task is unable to increase the size of the session cache data space.

The error codes have the following values:

**1 = DSPSERV EXTEND failed.**

The reason code contains the DSPSERV return code in the upper halfword and bits 8-23 of the DSPSERV reason code in the lower halfword.

**User Response:** The new session cache entry is not stored in the session cache data space. Refer to the DSPSERV description in *z/OS MVS Authorized Assembler Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01034E Unable to send cross-system message: Error *error-code*, Reason *reason-code*.**

**Explanation:** The GSKSRVR started task is unable to send a message to another member of the GSKSRVGP group.

The error codes have the following values:

- 1 = Unable to obtain XCF control lock on target system.
- 2 = Cross-system services are not available.
- 3 = Requested token not found on target system.
- 4 = User not authorized to access token data.
- 5 = Unable to allocate storage on the target system.
- 6 = Target replica is not a member of the GSKSRVGP group.
- 7 = Target replica is not active.
- 8 = IXCMMSGO failed. The reason code contains the IXCMMSGO return code in the upper halfword and the IXCMMSGO reason code in the lower halfword.
- 9 = IXCMMSGI failed on the target system. The reason code contains the IXCMMSGI return code in the upper halfword and the IXCMMSGI reason code in the lower halfword.
- 10 = Request function code is not supported.
- 11 = Request canceled.
- 12 = Unknown notification message.
- 13 = No response received from target system.
- 14 = Unable to allocate storage on the local system.
- 15 = IXCMMSGI failed on the local system. The reason code contains the IXCMMSGI return code in the upper halfword and the IXCMMSGI reason code in the lower halfword.

**User Response:** The request is not processed. Refer to the IXCMMSGI or IXCMMSGO description in *z/OS MVS Sysplex Services Reference* for more information. Contact your service representative if the error persists.

---

**GSK01035E SSL server is not available.**

**Explanation:** The SSL server task is not available. This error will occur if the GSKSRVR started task is not running, has not completed initialization, or is stopping.

**User Response:** Wait until the GSKSRVR started task is available and then retry the failing request.

---

**GSK01036E No job name specified.**

**Explanation:** No job name was specified on the TRACE CT command when starting a component trace.

**User Response:** Specify at least one job name when starting a component trace.

---

**GSK01037E Unable to call SSL server: Error *number*, Reason *number*.**

**Explanation:** The command processor for the TRACE CT command is unable to call the GSKSRVR started task.

The following error codes are defined:

- 8 = Parameter buffer overflow
- 12 = Unable to allocate storage
- 16 = Local service support is not enabled
- 20 = Program call task abended (the reason is the abend code)
- 24 = Unable to obtain control lock
- 28 = Requested function is not supported

**User Response:** Verify that the GSKSRVR started task is running on the local system. Contact your service representative if the error persists.

---

**GSK01038E Incorrect trace option specified.**

**Explanation:** The OPTIONS parameter on the TRACE CT command does not specify a valid SSL trace option. The only valid option is LEVEL=*n* where *n* is the requested SSL trace level. Refer to the description of the GSK\_TRACE environment variable for more information on setting the SSL trace level.

**User Response:** Specify a valid SSL trace option.

---

**GSK01039E The trace buffer size must be between 64K and 512K.**

**Explanation:** The trace buffer size specified on the TRACE CT command must be between 64K and 512K.

**User Response:** Specify a valid trace buffer size.

---

**GSK01040I SSL component trace started.**

**Explanation:** The SSL component trace has been started. The jobs specified on the TRACE CT command may be already running or may be started after the TRACE CT command is processed. However, any jobs that are already running must have been started after

the GSKSRVR started task was started.

**User Response:** None

---

**GSK01041I SSL component trace ended.**

**Explanation:** The SSL component trace has ended.

**User Response:** None

---

**GSK01042E Incorrect OPTIONS syntax**

**Explanation:** The OPTIONS parameter syntax on the IPCS CTRACE command is not correct for an SSL component trace. SSL supports three options: JOB, PID, and TID. The CTRACE OPTIONS parameter is specified as CTRACE COMP(GSKSRVR) OPTIONS((JOB(name),PID(hexid),TID(hexid))).

**User Response:** Specify a valid OPTIONS parameter.

---

**GSK01043E Incorrect trace option.**

**Explanation:** An incorrect trace option was specified on the IPCS CTRACE command for an SSL component trace. SSL supports three options: JOB, PID, and TID. The CTRACE OPTIONS parameter is specified as CTRACE COMP(GSKSRVR) OPTIONS((JOB(name),PID(hexid),TID(hexid))). The job name must be 1-8 characters. The hexadecimal identifier for PID and TID must be 1-8 hexadecimal digits.

**User Response:** Specify a valid OPTIONS parameter.

---

**GSK01044E Duplicate trace option.**

**Explanation:** An SSL trace option is specified more than once on the IPCS CTRACE command.

**User Response:** Do not specify the same trace option more than once.

---

**GSK01045E Incorrect hexadecimal value.**

**Explanation:** The value for the PID and TID trace options for the IPCS CTRACE command must be a hexadecimal value consisting of 1-8 hexadecimal digits.

**User Response:** Specify a valid hexadecimal value.

---

**GSK01046I Trace filter options: *option list***

**Explanation:** The IPCS CTRACE command specifies one or more trace entry filter options.

**User Response:** None

---

**GSK01047I SSL component trace started for *jobname*.**

**Explanation:** The SSL component trace has started for the indicated job. This message is displayed for each job specified on the TRACE CT command when

the application makes its first SSL API request after SSL component trace has been started.

**User Response:** None

---

## Utility Messages (GSK00nnn)

This section describes utility messages.

---

**GSK00001E Unable to open trace file *name*:**  
*error-message*

**Explanation:** The gsktrace command is unable to open the trace file.

**User Response:** Verify that the trace file exists and can be accessed by the user issuing the gsktrace command. Contact your service representative if the error persists.

---

**GSK00002E Unable to read trace file *name*:**  
*error-message*

**Explanation:** The gsktrace command is unable to read the trace file.

**User Response:** Verify that there are no filesystem errors and that the trace file has not been modified. Contact your service representative if the error persists.

---

**GSK00003E Trace record length *size* exceeds the maximum length.**

**Explanation:** A record in the trace file is longer than the maximum length for a trace record. This probably means the trace file has been modified.

**User Response:** Verify that the trace file has not been modified and was created by a compatible level of the System SSL runtime.

---

**GSK00004R Enter password:**

**Explanation:** The System SSL runtime is needs a database or certificate password.

**User Response:** Enter the requested password.

---

**GSK00005R Re-enter password:**

**Explanation:** The System SSL runtime is verifying the password.

**User Response:** Enter the same password you entered for the first password prompt.

---

**GSK00006E File *name* is not a valid SSL trace file.**

**Explanation:** The gsktrace command is unable to process the file because it is not in the proper format. This error can occur if the trace file was created by an earlier level of the System SSL runtime.

**User Response:** Process the trace file using the gsktrace command that is at the same level as the System SSL runtime which created the trace file.

---

**GSK00007R Enter new password:**

**Explanation:** The System SSL runtime is needs a new database password.

**User Response:** Enter the requested password.

## Appendix A. Environment Variables

The following tables contain all the environment variables used by System SSL.

Table 4. SSL-Specific Environment Variables

Environment Variables	Usage	Valid Values
GSK_CRL_CACHE_TIMEOUT	Specifies the number of hours that a cached CRL will remain valid.	The valid timeout values are 0 through 720 and defaults to 24. A value of 0 will disable the CRL cache.
GSK_HW_CRYPTO	<p>Specifies whether the hardware cryptographic support will be used. Note that ICSF (Integrated Cryptographic Service Facility) must be configured and running in order for System SSL to use the hardware cryptographic support.</p> <p>Selected hardware cryptographic functions can be disabled by setting the appropriate bits to zero in the GSK_HW_CRYPTO value. The corresponding software algorithms will be used when a hardware function is disabled. The following bit assignments are defined:</p> <ul style="list-style-type: none"> <li>1 = Not used</li> <li>2 = 56-bit DES encryption/decryption</li> <li>4 = 168-bit Triple DES encryption/decryption</li> <li>8 = Public key encryption/decryption</li> </ul>	A value of '0' will disable the use of the hardware support while a value of '65535' will enable the use of the hardware support. The hardware support will be used if this environment variable is not defined.
GSK_KEY_LABEL	Specifies the label of the key used to authenticate the application. The default key will be used if a key label is not specified.	
GSK_KEYRING_FILE	<p>Specifies the name of the key database HFS file or the SAF key ring. A key database is used if the GSK_KEYRING_PW or GSK_KEYRING_STASH environment variable is also specified. Otherwise a SAF key ring is used.</p> <p>The user must have READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by the user. The user must have UPDATE access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using a SAF key ring owned by another user.</p> <p>Note that certificate private keys are not available when using a SAF key ring owned by another user.</p>	The SAF key ring name is specified as "userid/keyring". The current userid is used if the userid is omitted.
GSK_KEYRING_PW	Specifies the password for the key database.	NULL or value consisting of up to 128 characters.

## Environment Variables

Table 4. SSL-Specific Environment Variables (continued)

Environment Variables	Usage	Valid Values
GSK_KEYRING_STASH	Specifies the name of the key database password stash file.	The stash file name always has an extension of ".sth" and the supplied name will be changed if it does not have the correct extension. The GSK_KEYRING_PW environment variable will be used instead of the GSK_KEYRING_STASH environment variable if it is also specified.
GSK_LDAP_SERVER	Specifies one or more blank-separated LDAP server host names. The LDAP server is used to obtain CA certificates when validating a certificate and the local database does not contain the required certificate. The local database must contain the required certificates if no LDAP server is specified. Even when an LDAP server is used, root CA certificates must be found in the local database since the LDAP server is not a trusted data source. The LDAP server is also used to obtain certificate revocation lists.	Each host name can contain an optional port number separated from the host name by a colon.
GSK_LDAP_PASSWORD	Specifies the password to use when connecting to the LDAP server.	
GSK_LDAP_PORT	Specifies the LDAP server port. Port 389 will be used no LDAP server port is specified.	Port must be between 1 and 65535.
GSK_LDAP_USER	Specifies the distinguished name to use when connecting to the LDAP server.	
GSK_PROTOCOL_SSLV2	Specifies whether the SSL V2 protocol is supported. The SSL V2 protocol should be disabled whenever possible since the SSL V3 protocol provides significant security enhancements.	A value of "0", "OFF" or "DISABLED" will disable the SSL V2 protocol while a value of "1", "ON" or "ENABLED" will enable the SSL V2 protocol.
GSK_PROTOCOL_SSLV3	Specifies whether the SSL V3 protocol is supported.	A value of "0", "OFF" or "DISABLED" will disable the SSL V3 protocol while a value of "1", "ON" or "ENABLED" will enable the SSL V3 protocol.
GSK_PROTOCOL_TLSV1	Specifies whether the TLS V1 protocol is supported.	A value of "0", "OFF" or "DISABLED" will disable the TLS V1 protocol while a value of "1", "ON" or "ENABLED" will enable the TLS V1 protocol.
GSK_STDERR_FILE	Specifies the fully-qualified name of the file to receive standard error messages generated using SSL message services. Messages will be written to stderr if this environment variable is not defined.	

Table 4. SSL-Specific Environment Variables (continued)

Environment Variables	Usage	Valid Values
GSK_STDOUT_FILE	Specifies the fully-qualified name of the file to receive standard output messages generated using SSL message services. Messages will be written to stdout if this environment variable is not defined.	
GSK_SYSPLEX_SIDCACHE	Specifies whether sysplex session caching is supported for this application.	A value of "0", "OFF" or "DISABLED" will disable sysplex session caching while a value of "1", "ON" or "ENABLED" will enable sysplex session caching.
GSK_TRACE	Specifies a bit mask enabling System SSL trace options. No trace option is enabled if the bit mask is 0 and all trace options are enabled if the bit mask is 0xffff. The bit mask can be specified as a decimal (nnn), octal (0nnnn) or hexadecimal (0xhh) value.	The following trace options are available: 0x01 = Trace function entry 0x02 = Trace function exit 0x04 = Trace errors 0x08 = Include informational messages 0x10 = Include EBCDIC data dumps 0x20 = Include ASCII data dumps
GSK_TRACE_FILE	Specifies the name of the trace file and defaults to /tmp/gskssl.%trc. The gsktrace command is used to format the trace file. The trace file is not used if the GSK_TRACE environment variable is not defined or is set to 0.  The current process identifier is included as part of the trace file name when the name contains a percent sign (%). For example, if GSK_TRACE_FILE is set to /tmp/gskssl.%trc and the current process identifier is 247, then the trace file name will be /tmp/gskssl.247.trc.	Must be set to the name of an HFS file in a directory for which the executing application has write permission.

## Environment Variables

Table 4. SSL-Specific Environment Variables (continued)

Environment Variables	Usage	Valid Values
GSK_V2_CIPHER_SPECS	<p>Specifies the SSL V2 cipher specifications in order of preference as a string consisting of 1 or more 1-character values. The following cipher specifications are supported:</p> <ul style="list-style-type: none"> <li>1 = 128-bit RC4 encryption with MD5 message authentication (128-bit secret key)</li> <li>2 = 128-bit RC4 export encryption with MD5 message authentication (40-bit secret key)</li> <li>3 = 128-bit RC2 encryption with MD5 message authentication (128-bit secret key)</li> <li>4 = 128-bit RC2 export encryption with MD5 message authentication (40-bit secret key)</li> <li>6 = 56-bit DES encryption with MD5 message authentication (56-bit secret key)</li> <li>7 = 168-bit Triple DES encryption with MD5 message authentication (168-bit secret key)</li> </ul>	<p>The default is "713642" if security level 3 is installed, "642" otherwise.</p>
GSK_V2_SESSION_TIMEOUT	<p>Specifies the session timeout value in seconds for the SSL V2 protocol.</p>	<p>The valid timeout values are 0 through 100, default value is 100.</p>
GSK_V2_SIDCACHE_SIZE	<p>Specifies the number of session identifiers that can be contained in the SSL V2 cache.</p>	<p>The valid cache sizes are 0 through 32000 and defaults to 256. The SSL V2 cache will be disabled if 0 is specified.</p>

Table 4. SSL-Specific Environment Variables (continued)

Environment Variables	Usage	Valid Values
GSK_V3_CIPHER_SPECS	<p>Specifies the SSL V3 cipher specifications in order of preference as a string consisting of 1 or more 2-character values. The SSL V3 cipher specifications are used for the SSL V3 and TLS V1 protocols. The following cipher specifications are supported:</p> <ul style="list-style-type: none"> <li>00 = No encryption or message authentication and RSA key exchange</li> <li>01 = No encryption with MD5 message authentication and RSA key exchange</li> <li>02 = No encryption with SHA-1 message authentication and RSA key exchange</li> <li>03 = 40-bit RC4 encryption with MD5 message authentication and RSA key exchange</li> <li>04 = 128-bit RC4 encryption with MD5 message authentication and RSA key exchange</li> <li>05 = 128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange</li> <li>06 = 40-bit RC2 encryption with MD5 message authentication and RSA key exchange</li> <li>09 = 56-bit DES encryption with SHA-1 message authentication and RSA key exchange</li> <li>0A = 168-bit Triple DES encryption with SHA-1 message authentication and RSA key exchange</li> <li>2F = 128-bit AES encryption with SHA-1 message authentication and RSA key exchange</li> <li>35 = 256-bit AES encryption with SHA-1 message authentication and RSA key exchange</li> </ul>	<p>The default is "0504352F0A090306020100" if security level 3 is installed, "090306020100" otherwise.</p>
GSK_V3_SESSION_TIMEOUT	<p>Specifies the session timeout value in seconds for the SSL V3 and TLS V1 protocols.</p>	<p>The valid timeout values are 0 through 86400 and defaults to 86400. The timeout will be disabled if 0 is specified.</p>
GSK_V3_SIDCACHE_SIZE	<p>Specifies the number of session identifiers that can be contained in the SSL V3 cache. The SSL V3 session cache is used for the SSL V3 and TLS V1 protocols.</p>	<p>The valid cache sizes are 0 through 64000 and defaults to 512. The SSL V3 cache will be disabled if 0 is specified.</p>
GSKV2CACHESIZE	<p>Used to control the size limit for a V2 session cache. This variable is for use only with the deprecated API set.</p>	<p>The valid cache sizes are 0 through 32000 and defaults to 256.</p>

## Environment Variables

Table 4. SSL-Specific Environment Variables (continued)

Environment Variables	Usage	Valid Values
GSKV3CACHESIZE	Used to control the size limit for a V3 session cache. This variable is for use only with the deprecated API set.	The valid cache sizes are 0 through 64000 and defaults to 512 entries.

Table 5 on page 323 contains system environment variables used by SSL. For more information, see the section on shell variables in the *z/OS: UNIX System Services Command Reference*.

Table 5. System Environment Variables used by SSL

System Environment Variables	Usage	Valid Values
LIBPATH	Used to specify the directory to search for a DLL (Dynamic Link Library) filename. If it is not set, the working directory is searched.	
NLSPATH	Specifies where the message catalogs are to be found.	
PATH	Contains a list of directories that the system searches to find executable commands. Directories in this list are separated with colons. Searches each directory in the order specified in the list until it finds a matching executable. If you want the shell to search the working directory, put a null string in the list of directories (for example, to tell the shell to search the working directory first, start the list with a colon or semicolon).	
STEPLIB	Identifies a STEPLIB variable to be used in building a process image for running an executable file. A STEPLIB is a set of private libraries used to store a new or test version of an application program, such as a new version of a runtime library.	STEPLIB can be set to the values CURRENT or NONE or to a list of MVS data set names. The default is CURRENT, which passes on the TASKLIB, STEPLIB, or JOBLIB allocations that are part of the invoker's MVS program search order environment to the process image created for an executable file. The value NONE indicates you do not want a STEPLIB environment for executable files. You can specify up to 255 MVS data set names, separated by colons, as a list of data sets used to build a STEPLIB variable.

## Environment Variables

---

## Appendix B. Sample C++ SSL Files

A sample set of files is shipped to provide an example of what is needed to build a C++ System SSL application. These files build one DLL (SECURES) and two programs: **client** and **server**. These sample files are located in `/usr/lpp/gskssl/examples`:

- Makefile
- client.cpp
- server.cpp
- common.hpp
- common.cpp
- secures.h
- secures.cpp
- utils.hpp
- utils.cpp
- display\_certificate.c

**server** (source file: `server.cpp`) is a multithreaded program that opens a socket on IP address 127.0.0.1, port 4321 and listens for client requests. **server** can run in either secure (using SSL) mode or nonsecure (using normal socket reads and writes) mode. By default, **server** runs with one socket listen thread and 20 work threads. The socket listen thread listens for connections from clients and puts each request onto the work list. The work threads check the work list for work and then perform the work. The number of work threads can be specified using the `-numthreads` parameter when starting **server**.

**client** (source file: `client.cpp`) is a single threaded program that connects to the server program and exchanges one or more data packets. **client** can also run in secure or nonsecure mode, but its mode must match the mode of the server to which it is connecting. The number of connections, the number of read/write packets per connection, the number of bytes in each write packet, and the number of bytes in each read packet can be specified. Multiple clients can be run simultaneously to the same server.

The files included in the examples are:

### Makefile

This file builds the example DLL and programs.

### client.cpp

This file contains the routines that implement the client function.

### server.cpp

This file contains the routines that implement the server function.

### common.hpp

This contains the prototypes and defines for the routines in `common.cpp`.

### common.cpp

This file contains a set of routines called by `client` and `server` to set up, accept, open, and close connections, and to read and write data. All data that is read or written in the form of packets that contain a header containing a command, length, and cookie. This implements a higher level communication protocol used between the `client` and `server` programs. For example, this higher level protocol allows the `client` to send a "STOP" request to the `server`, which stops the `server` program.

### secures.h

This file contains prototypes and defines for the routines in `secures.cpp`.

## Sample C++ SSL Files

secures.cpp

This file implements a set of APIs that are similar to the normal sockets APIs, except that the routines work in either secure (SSL) or nonsecure mode. These routines are called by code in client.cpp, server.cpp, and common.cpp.

utils.hpp

This file contains the prototype for the routine in utils.cpp, some structure definitions, and several defined constants.

utils.cpp

This file contains routines that server and client programs use to check command line options.

| display\_certificate.c

| This file is a sample program to decode and display an X.509 certificate.

---

## Appendix C. Sample Java SSL Files

A set of sample files is shipped, which provide an example of what is needed to build a z/OS System SSL Java application. This example builds two programs, **JavaClient** and **JavaServer**. These sample files are located in `/usr/lpp/gskssl/examples/java`.

JavaClient (source file: JavaClient.java) is a single threaded client that connects to the server program and exchanges a few packets of information. The client can run in either secure or non-secure mode, but its mode must match the mode of the server to which it is connecting. The number of connections, number of reads and writes per connection, and the number of bytes per read/write can all be specified.

JavaServer (source file: JavaServer.java) is a multi-threaded program that opens a socket and listens for client requests. The server, similar to the client, can run in secure and non-secure modes. The socket listen thread listens for connections from clients and forwards the connection to a worker thread. The number of threads is equal to the number of connections that are received from any number of clients.

The files included in the examples are:

### **Makefile**

This file builds the example DLL programs.

### **JavaServer.java**

This file contains the routines that implement the server functionality.

### **JavaClient.java**

This file contains the routines that implement the client functionality.

## Sample Java SSL Files

---

## Appendix D. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size.

---

### Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

---

### Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, SA22-7787, *z/OS TSO/E User's Guide*, SA22-7794, and *z/OS ISPF User's Guide Volume I*, SC34-4822, for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.



---

## Appendix E. 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 give 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, Japan

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

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

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

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

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

IBM Corporation  
Mail Station P300  
2455 South Road  
Poughkeepsie, NY 12601-5400  
U.S.A

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

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

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 illustrates 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. 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 IBM's application programming interfaces.

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

This product contains code licensed from RSA Security, Inc.



---

## Programming Interface Information

*z/OS: System Secure Sockets Layer Programming, SC24-5901* primarily documents intended Programming Interfaces that allow the customer to write programs to obtain services of System SSL.

*z/OS: System Secure Sockets Layer Programming, SC24-5901* also documents information that is NOT intended to be used as Programming Interfaces of System SSL. This information is identified where it occurs, by an introductory statement to a chapter or section.

---

## Trademarks

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

BookManager  
OS/390

IBM  
RACF

OS/2  
WorldRegistry

Domino is a trademark of Lotus Development Corporation in the United States, or other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation.

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



---

## Bibliography

This bibliography provides a list of some of the publications that are useful when using the z/OS System SSL support. The complete title, order number, and a brief description is given for each publication.

---

### z/OS Security Server Publications

- *z/OS: SecureWay Security Server LDAP Client Programming*, SC24-5924

This book describes the Lightweight Directory Access Protocol (LDAP) client APIs that you can use to write distributed applications on z/OS DCE and gives you information on how to develop LDAP applications.

- *z/OS: Security Server LDAP Server Administration and Use*, SC24-5923

This book describes how to install, configure, and run the stand-alone LDAP daemon (SLAPD). It is intended for administrators who will maintain the server and database.

- *z/OS: Security Server DCE Overview*, GC24-5921

This book describes the DCE security server and provides a road map for DCE security server information in the z/OS DCE library.

- *z/OS: Security Server RACF Security Administrator's Guide*, SA22-7683

This book explains RACF concepts and describes how to plan for and implement RACF.

- *z/OS: Security Server RACF Command Language Reference*, SA22-7687

This book describes the functions and syntax of all RACF commands.

---

### z/OS Cryptographic Services Publications

- *z/OS: Open Cryptographic Services Facility Application Programming*, SC24-5899

This book describes an overview of the Open Cryptographic Services Facility (OCSF). It explains how to integrate OCSF into applications and contains a sample OCSF application. It also defines the interfaces that application developers employ to access security services by the OCSF Framework and service provider modules. Specific information about the individual service providers is described.

- *z/OS: Open Cryptographic Services Facility Service Provider Module Developer's Guide and Reference*, SC24-5900

This book describes the features common to all OCSF service provider modules. It defines the interfaces for certificate, trust and data library service providers. Service provider developers must conform to these interfaces in order for the individual service provider modules to be accessible through the OCSF Framework.

---

### IBM C/C++ Language Publication

- *z/OS: C/C++ Programming Guide*, SC09-4765

This book describes how to develop applications in the C/C++ language in OS/390.

---

### Other IBM z/OS Publications

- *z/OS: Information Roadmap*, SA22-7500

This book lists the complete titles and order numbers of the books for all products that are part of z/OS.

- *z/OS: Routing and Descriptor Codes*, SA22-7624

This book contains routing and descriptor codes for messages.

- *z/OS: Program Directory*, GI10-0669

The Program Directory contains information about installing the Cryptographic Services base element using SMP/E.

- *z/OS: MVS Programming: Assembler Services Reference, Volume 1*, SA22-7606 and *z/OS: MVS Programming: Assembler Services Reference, Volume 2*, SA22-7607

These books describe Assembler services information.



# Index

## A

- accepting a secure socket connection 233
- accessibility 329
- accessing DLLs 250
- APIs
  - `gsk_attribute_get_buffer()` 29
  - `gsk_attribute_get_cert_info()` 31
  - `gsk_attribute_get_data()` 34
  - `gsk_attribute_get_enum()` 36
  - `gsk_attribute_get_numeric_value()` 38
  - `gsk_attribute_set_buffer()` 40
  - `gsk_attribute_set_callback()` 43
  - `gsk_attribute_set_enum()` 47
  - `gsk_attribute_set_numeric_value()` 49
  - `gsk_environment_close()` 51
  - `gsk_environment_init()` 52
  - `gsk_environment_open()` 54
  - `gsk_free_cert_data()` 58
  - `gsk_free_memory()` 226
  - `gsk_get_cert_by_label()` 59
  - `gsk_get_cipher_info` 227
  - `gsk_get_dn_by_label()` 228
  - `gsk_get_update()` 63
  - `gsk_initialize()` 229
  - `gsk_list_free()` 64
  - `gsk_secure_soc_close()` 232
  - `gsk_secure_soc_init()` 233
  - `gsk_secure_soc_read()` 238
  - `gsk_secure_soc_reset()` 240
  - `gsk_secure_soc_write()` 241
  - `gsk_secure_socket_close()` 65
  - `gsk_secure_socket_init()` 66
  - `gsk_secure_socket_misc()` 69
  - `gsk_secure_socket_open()` 71
  - `gsk_secure_socket_read()` 72
  - `gsk_secure_socket_write()` 76
  - `gsk_uninitialize()` 246
  - `gsk_user_set()` 247
  - GSKSRBRD() 244
  - GSKSRBWT() 245, 251
  - using in an System SSL program 5

## B

- bibliography 335
- building a Java application 25
- building a z/OS Java application 26
- building a z/OS System SSL application 11
- building an System SSL application 16

## C

- callback routine for IO 19
- certificate
  - removing 275
  - self-signed, creating 264
- certificate management 249

- cipher information
  - querying 227
- client, authentication certificate selection 18
- client, System SSL program 14
- compiling an System SSL application 16
- creating
  - SSL environment 11

## D

- diagnostic information 287
- disability 329
- distinguished name
  - returning pointer for 228
- DLLS, accessing 250

## E

- elements of an System SSL program 5
- ending secure socket connection 232
- environment variables 317
- establishing System SSL environment 229
- examples
  - parts shipped in HFS 2

## F

- FMID
  - Cryptographic Services Security Level 3 1
  - Cryptographic Services System SSL 1
  - Japanese 1

## G

- `gsk_attribute_get_buffer()` 29
- `gsk_attribute_get_cert_info()` 31
- `gsk_attribute_get_data()` 34
- `gsk_attribute_get_enum()` 36
- `gsk_attribute_get_numeric_value()` 38
- `gsk_attribute_set_buffer()` 40
- `gsk_attribute_set_callback()` 43
- `gsk_attribute_set_enum()` 47
- `gsk_attribute_set_numeric_value()` 49
- `gsk_environment_close()` 51
- `gsk_environment_init()` 52
- `gsk_environment_open()` 54
- `gsk_free_cert_data()` 58
- `gsk_free_memory()` API 226
- `gsk_get_cert_by_label()` 59
- `gsk_get_cipher_info()` API 227
- `gsk_get_dn_by_label()` API 228
- `gsk_get_update()` 63
- `gsk_initialize()` API 229
- `gsk_list_free()` 64
- `gsk_secure_soc_close` API 232
- `gsk_secure_soc_init()` API 233
- `gsk_secure_soc_read()` API 238

- gsk\_secure\_soc\_reset() API 240
- gsk\_secure\_soc\_write() API 241
- gsk\_secure\_socket\_close() 65
- gsk\_secure\_socket\_init() 66
- gsk\_secure\_socket\_misc() 69
- gsk\_secure\_socket\_open() 71
- gsk\_secure\_socket\_read() 72
- gsk\_secure\_socket\_write() 76
- gsk\_soc\_init\_data structure 233
- gsk\_uninitialize() API 246
- gsk\_user\_set() API 247
- gskkyman utility
  - accessing DLLs 250
  - being your own certificate authority 279
  - certificate, self signed
    - creating 264
  - certificates
    - removing 275
  - HFS location 2
  - private key
    - removing 275
  - setting LANG environment variable 250
  - setting NLSPATH environment variable 250
  - setting PATH environment variable 250
  - setting STEPLIB environment variable 250
  - setting up the environment 250
  - using 249
- GSKSRBRD() 244
- GSKSRBWT() 245, 251
- gskssl.h header file
  - gsk\_soc\_init\_data structure 233
  - HFS location 2

## H

- handshake process 233
- header file, gskssl.h
  - See gskssl.h header file 2
- HFS
  - contents 2
  - parts shipped 2

## I

- initializing data areas for System SSL 233
- initiating a secure socket connection 233
- installation information 2
- installation PDS
  - members of 3
  - name of 1

## K

- key database file
  - reading 229
  - uninitialize 246
- key management 249
- key ring, definition of 249
- keyboard 329

## L

- LANG environment variable, setting 250

## M

- managing PKI private keys and certificates 249
- migrating to the new SSL interfaces 23

## N

- NLSPATH environment variable, setting 250

## O

- obtaining System SSL trace information 287
- organization of book xi

## P

- PATH environment variable, setting 250
- PDS
  - identified in STEPLIB 16
- PDS, installation
  - members of 3
  - name of 1
- private keys
  - removing 275
- programming interfaces
  - using in an System SSL program 5

## Q

- querying cipher information 227

## R

- RACDCERT command 249
- RACF key ring
  - reading 229
  - uninitialize 246
- receiving data on secure socket connection 238
- refreshing security parameters 240
- removing
  - certificate/private key from key database 275
- removing settings for the System SSL
  - environment 246
- returning distinguished name 228
- running a Java application 25
- running a z/OS Java application 26
- running an System SSL Application 16

## S

- sample files
  - list of 325
- sample Java files
  - list of 327
- secure socket connection
  - accepting 233

- secure socket connection (*continued*)
  - ending 232
  - initiating 233
  - receiving data 238
  - sending data 241
- Secure Sockets Layer (SSL)
  - See SSL (Secure Sockets Layer) 1
- sending data on secure socket connection 241
- server, System SSL program 12
- session ID (SID) 20
- setting
  - gskkyman environment 250
  - LANG environment variable 250
  - NLSPATH environment variable 250
  - PATH environment variable 250
  - STEPLIB environment variable 250
- shortcut keys 329
- softcopy publications xiii
- software dependencies 1
- SSL (Secure Sockets Layer)
  - description 1
- SSL environment
  - creating 11
- SSL System
  - callback routine for IO 19
- STEPLIB environment variable, setting 250
- structure
  - gsk\_soc\_init\_data 233
- structure of an System SSL program 5
- System SSL
  - APIs 27
  - client authentication certificate selection 18
  - elements of a program 5
  - environment variables 317
  - establishing environment 229
  - how it works 5
  - migrating 23
  - obtaining trace information 287
  - parts shipped in HFS 2
  - parts shipped in PDS 3
  - refreshing security parameters 240
  - removing settings for the environment 246
  - session ID (SID) cache replacement 19
  - using hardware cryptographic features 9
  - writing a Java source program 25
- System SSL application
  - building 16
  - building a z/OS Java application 26
  - building and running a Java application 25
  - overview 6
  - running a z/OS Java application 26
  - writing a client program 14
  - writing a server program 12
  - writing a source program 11
  - writing and building 11

## W

- writing
  - Java source program 25
  - system SSL client program 14
  - system SSL server program 12
  - system SSL source program 11
  - z/OS System SSL application 11

## U

- using hardware cryptographic features with System SSL 9







Program Number: 5694-A01

Printed in U.S.A.

SC24-5901-02



Spine information:



z/OS

System SSL Programming VIR4.0