IBM Global Security Kit

GSKCapiCmd User's Guide

GSKit Version 7

Edition 12 March 2007

Note: Before using this information and the product it supports, read the information in Appendix 3. Notices.

Contents

Preface

This manual is intended for network or system security administrators who have installed GSKit and want to use the GSKCapiCmd program to modify CMS or PKCS11 key databases. This manual assumes the reader is familiar with the GSKit product range and the functionality of the CMS key database.

Before continuing to read this manual ensure you have read and understood the following prerequisite readings. This will ensure that you understand the required concepts and terms used throughout the manual:

- Appendix 1 of this manual, “CMS Key Databases”.
- Appendix 2 of this manual, “A Simple Example”.

Please ensure that you read all of the identified readings before you continue with this manual.

How this book is organized

This manual contains the following sections:

- **Chapter 1**, “Using the GSKCapiCmd Program”, on page 6. This chapter contains general information about the GSKCapiCmd program.
- **Chapter 2**, “Key Database Commands”, on page 8. This chapter looks at the different commands that are available to the GSKCapiCmd program for CMS key databases.
- **Chapter 3**, “Certificate Commands”, on page 17. This chapter looks at the different commands that are available to the GSKCapiCmd program for managing certificates.
- **Chapter 4**, “Certificate Request Commands” on page 38. This chapter looks at the different commands that are available to the GSKCapiCmd program for managing certificate requests.
- **Chapter 5**, “Random Commands”, on page 48.
- **Chapter 6** “Help Command”, on page 50.
- **Chapter 7** “Version Command”, on page 51.
- **Chapter 8** “Error Codes and messages”, on page 52.
- **Appendix 1**, “CMS Key Databases”, on page 58.
- **Appendix 2,** “A Simple Example”, on page 61
- **Appendix 3,** “Notices”, on page 67

Contacting software support

Before contacting IBM Tivoli Software Support with a problem, refer to the IBM Tivoli Software Support site by clicking the Tivoli support link at the following Web site: http://www.ibm.com/software/support/ If you need additional help, contact software support by using the methods described in the *IBM Software Support Guide* at the following Web site: http://techsupport.services.ibm.com/guides/handbook.html The guide provides the following information: Registration and eligibility requirements for receiving support v Telephone numbers, depending on the country in which you are located v A list of information you should gather before contacting customer support

Conventions used in this book

The following typeface conventions are used through out this manual:

Bold: Draws emphasis to a key word, indicating that the user needs to take note as the word will be used at a later stage. In relation to the options of each command a bolded value indicates the default value for that option.

Italic: Non-specific command line options or identifiers.

Symbol conventions

[ ] - Identifies an option that is optional, if an option is not surrounded by this style of brackets the option is required.

| - Indicates an “OR” relationship between the options on either side of it.

{} – Identifies mutually exclusive set of options.

Operating system differences

This book uses the UNIX convention for specifying environment variables and for directory notation. When using the Windows command line, replace *$variable* with *%variable%* for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Acronyms

The following is a list of acronyms that are used throughout this manual.

CA	-	Certificate Authority
CMS	-	Certificate Management System
FIPS	-	Federal Information Processing Standards
GSKit	-	Global Security Kit
ICC	-	IBM Crypto for C
IE	-	Internet Explorer

Revision History

Revision Description	On Date	Refer to...
First draft of document	11/08/2004	Anthony Ferguson
Updating document with additional information	24/08/2004	Anthony Ferguson
Updating the document with additional information.	3/11/2004	Anthony Ferguson
Corrected a number of error in the document after a review.	2/12/2004	Anthony Ferguson
Updated a number of errors after an @sec review	21/12/2004	Anthony Ferguson
Adding the random password generation information	7/1/2005	Anthony Ferguson
Correcting some minor errors.	24/01/2005	Anthony Ferguson
Adding important warning to ensure quotes are used for all commands	14/02/2005	Anthony Ferguson
Addressing atsec comments	28/02/2005	Alex Hennekam
Adding the email and dc	11/08/2005	Anthony Ferguson

attributes to the DN tags		
Adding the –sernum tag to the certificate sign command	25/08/2005	Anthony Ferguson
Adding the –expiry action	16/02/2006	Anthony Ferguson
Adding pfx documentation for certificate import and export commands	10/04/2006	Anthony Ferguson
Adding additional error codes to Chapter 8	27/04/2006	Anthony Ferguson
Adding newly supported SHA algorithms	29/06/2006	Anthony Ferguson
Updating required fields for DN tags	10/07/2006	Kai Gorman
Adding error codes 232 and 233	24/07/2006	Kai Gorman
Some corrections to argument lists	12/March/2007	Simon McMahon

Chapter 1. Using The GSKCapiCmd Program

GSKCapiCmd is a tool that can be used to manage keys, certificates and certificate requests within a CMS key database. The tool has all of the functionality that GSKit's existing java command line tool has except that GSKCapiCmd supports CMS and PKCS11 key databases. If you are intending to manage key databases other than CMS or PKCS11 you will need to use the existing java tool.

GSKCapiCmd can be used to manage all aspects of a CMS key database. The following chapters go into detail for each of the functions supported by GSKCapiCmd.

The advantage of using this tool over the existing one is that GSKCapiCmd does not require java to be installed on the system.

GSKCapiCmd uses some encoding rules, and implements aspects of certain RFC's and standards. While it is not strictly necessary for users to have a full understanding of these items in order to use this utility, those wishing to learn more should examine the following resources:

http://asn1.elibel.tm.fr/en/standards/index.htm

Basic Encoding Rules (BER)

BER encoding is defined in the specification ITU-T Rec. X.690 (2002).

Distinguished Encoding Rules (DER)

DER encoding is defined in the specification ITU-T Rec. X.690 (2002).

http://www.faqs.org/rfcs

PKCS#10

RFC 2986: PKCS #10: Certification Request Syntax Specification, Version 1.7, November 2000

X.509

RFC 3280: Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL), obsoletes RFC 2459, April 2002.

http://www.rsasecurity.com/rsalabs/

PKCS#11

Cryptographic Token Interface Standard

PKCS#12

PKCS 12 v1.0: Personal Information Exchange Syntax, RSA Laboratories, June 24, 1999

PKCS#7

PKCS 7 v1.5: Cryptographic Message Syntax, RSA Laboratories, March 1998

CMS

Appendix 1 offers some additional information concerning the format and use of a CMS keystore, while appendix 2 takes the reader though a simple example of how CMS keystores can be used to enable SSL communication between a server and client application.

GSKCapiCmd command line syntax

The syntax for the GSKCapiCmd program is as follows:

gsk7capicmd <*object*> <*action*> <*options*>

where:

object Is one of the one of the following:

-keydb
Actions acted on a key database.

-cert
Actions acted on a certificate stored within an identified key database.

-certreq
Actions acted on a certificate request stored within an identified key database.

-random
Generates a random string of characters that can be used as a password for other commands.

-version
Displays version information for GSKCapiCmd

-help
Displays help for the GSKCapiCmd commands.

action Is the specific action to be taken on the object.

options Are the options associated with the specified object and task

This manual will go into detail for each particular object, its associated actions, and what options are available in the following chapters.

Chapter 2. Key Database Commands

The key database commands are associated with the -keydb object. This object supports the following actions:

- Create a Key Database (-create)
- Delete a Key Database (-delete)
- Change the Password of an existing Key Database (-changepw)
- Stash the Password of an existing Key Database (-stashpw)
- List the Supported Key Databases (-list)
- Convert an old style CMS Key Database to the updated style of CMS Key Database (-convert)
- Display the expiry date associated with a CMS key databases password (-expiry)

Each of the following sections goes into detail on how to use each of the key database commands and what options are available for each command.

Create Key Database

The create command creates a new CMS key database. During the creation process three files are created. The first file is the certificate key database itself. By standard it is a good idea to name this file with a .kdb extension (key.kdb). This is not required, but it is a good idea, it makes easy to identify the file as a key database.

The second file created is used to store certificate requests associated with the key database. This file is created with the same name as given to the key database but with a .rdb extension. The third file is used to hold the certificate revocation list used by the key database; this file has become obsolete and is no longer used. This file is created with the same name as the key database but this time with a .crl extension.

Once the key database has been created it is populated with a number of pre-defined trusted certificate authority (CA) certificates. There is no way of preventing this from occurring. The trusted CA certificates are as follows:

Entrust.net Global Secure Server Certification Authority
Entrust.net Global Client Certification Authority
Entrust.net Client Certification Authority
Entrust.net Certification Authority (2048)
Entrust.net Secure Server Certification Authority
VeriSign Class 3 Public Primary Certification Authority
VeriSign Class 2 Public Primary Certification Authority
VeriSign Class 1 Public Primary Certification Authority
VeriSign Class 4 Public Primary Certification Authority - G2
VeriSign Class 3 Public Primary Certification Authority - G2
VeriSign Class 2 Public Primary Certification Authority - G2
VeriSign Class 1 Public Primary Certification Authority - G2
VeriSign Class 4 Public Primary Certification Authority - G3
VeriSign Class 3 Public Primary Certification Authority - G3
VeriSign Class 2 Public Primary Certification Authority - G3
VeriSign Class 1 Public Primary Certification Authority - G3
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA

Any or all of these CA certificates can be removed from the key database. If you want to remove any of the certificates look at the delete certificate command in this manual.

The syntax for creating a CMS key database with GSKCapiCmd is as follows:

gsk7capicmd -keydb -create -db <name> [-pw <passwd>] [-type <cms>] [-expire <days>] [-stash] [-fips] [-strong]

Where:

object -keydb

action -create

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘””, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
Fully qualified path name of a key database. A good example of a key database filename might be /home/<user name>/keydb.db.

-pw <passwd>
The password for the key database identified by the –db tag above. If you want to create a key store without a password simply leave the -pw tag out of the above command.

-type <cms>
The type of the key database to be created. At this stage this tool only supports the creation of CMS key database. If the tag is left off the tool will assume that a CMS key database is to be created.

-expire <days>
The number of days before the password for the key database is to expire. If the tag is left off, the key database password will never expire. If specified the duration must be from 1 to 7300 days (20 years).

-stash

Stash the password for the key database after creation. A stash file is used as an automatic way of providing a password. When accessing a key database the system will first check for the existence of a stash file. If one exists the contents of the file will be decrypted and used as input for the password. When the -stash option is specified during the create action, the password is stashed into a file with the name as follows: <key database name>.sth.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk*7*capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated.

-strong

Check that the password entered satisfies the minimum requirements for the passwords strength. The minimum requirements for a password are as follows:

- The minimum password length is 14 characters.
- A password needs to have at least one lower case character, one upper case character, and one digit or special character (eg. *$#% etc, a space is classified as special characters).
- Each character should not occur more than three times in a password.
- No more than two consecutive characters of the password should be identical.
- All characters mentioned above are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

Delete Key Database

The delete key database command simply deletes the identified key database. When identifying the key database you simply need to specify the file name of the key database. The request database (.rdb) and certificate revocation list (.crl) files are removed automatically during the process. If a stash file was created it is not removed.

If a password was provided for this command it is used to ensure that the user is actually allowed to delete the key database. If the password is not correct the key database is not deleted.

The syntax for deleting a key database with GSKCapiCmd is as follows:

gsk*7*capicmd -keydb -delete -db <name> [-pw <passwd>]

Where:

object -keydb

action -delete

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-pw <passwd>
The password for the key database identified by the –db tag above. The –pw tag is required if the key database was created with a password. It is an additional check to ensure that the user deleting the key database is authorized to do so. If the key database does not have a password the –pw tag is not required.

If a password is provided and it does not patch the password for the identified key database the key database is not deleted.

Change Password for Key Database

The change password command allows the user to change the password associated with the specified key database. When changing the password for a key database, all key records containing encrypted private key information have the private key data re-encrypted. The new password is used as input to create the new encryption key used during the encryption process.

The syntax for changing the password of an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -keydb -changepw {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -new_pw <new passwd> [-expire <days>] [-stash] [-fips] [-strong]

Where:

object -keydb

action -changepw

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db tag above. If you want to create a key store without a password simply leave the -pw tag out of the above command.

-new_pw <new passwd>
The new password for the key database.

-expire <days>
The number of days before the new password is to expire. If the tag is not specified the key databases password never expires. If specified the duration must be within the range of 1 to 7300 days (20 years).

-stash
Stash the password for the key database. When the –stash the new password will be stashed in a file with the filename built as follows: <filename of key database>.sth.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

-strong
Check that the password entered satisfies the minimum requirements for the passwords strength. The minimum requirements for a password are as follows:

- The minimum password length is 14 characters.
- A password needs to have at least one lower case character, one upper case character, and one digit or special character (eg. *$#% etc, a space is classified as special characters).
- Each character should not occur more than three times in a password.
- No more than two consecutive characters of the password should be identical.
- All characters mentioned above are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

Stash the Password for Key Database

The stash password command takes an existing key databases password and stashes it to a specified file. The reason that a user would want to stash a password for a key database is to allow the password to be recovered from the file when automatic login is required. The output of the command is a single file with the name of the key database with a ".sth" extension.

The syntax for stashing the password of an existing key database with GSKCapiCmd is as follows:

gsk*7*capicmd -keydb -stashpw -db <name> [-pw <passwd>] [-fips]

Where:

object -keydb

action -stashpw

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (""). You will also need to escape, using a '\' character, the following characters if they appear in the string values: '!', '\', '"', '`'. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db "/tmp/key.kdb" –pw "j\!jj"). Note however when prompted by gsk*7*capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-pw <passwd>
The password for the key database identified by the –db tag above.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will

be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

List Supported Key Databases

The list supported key databases simply lists all of the key database types that the GSKCapiCmd supports.

The syntax for listing the key databases supported by GSKCapiCmd is as follows:

gsk7capicmd -keydb –list

Where:

object -keydb

action -list

Convert Key Database

The convert key database command converts an old version CMS key database to the new version of CMS key database. The latest version of CMS is more secure because it uses more secure algorithms to secure the contents of the key databases during creation.

This command requires that you assign a name to the new key database that is different to the existing old key database. This name cannot be the same as the existing one. This is to ensure that the old key database is not destroyed until the user destroys it. Once all testing of the new version key database has been completed the user can remove the old key database and rename the new key database to the old key databases name if required.

The syntax for converting a key database to the latest CMS version by GSKCapiCmd is as follows:

gsk7capicmd -keydb –convert –db <name> [-pw <passwd>] –new_db <name> [-new_pw <passwd>] [-preserve] [-strong] [-fips]

Where:

object -keydb

action -list

options

Deleted: . At this stage this command will only indicate that CMS and PKCS11 key databases are supported.

Deleted: [-fips]

Deleted: ¶
options¶
¶
-fips¶
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail. ¶
¶
* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated¶

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>

The fully qualified path name of a key database.

-pw <passwd>

The password for the key database identified by the –db tag above.

-new_db <filename>

Fully qualified path name of a new key database to be created during the conversion.

-new_pw <passwd>

The password for the key database identified by the –new_db tag above.

-preserve

When this option is specified during the convert command the newly created key database will not include any new trusted CA certificates, it will be identical to the old key database.

If you chose not to include this option the following CA certificates will be added to the newly created key database:

Entrust.net Global Secure Server Certification Authority
Entrust.net Global Client Certification Authority
Entrust.net Client Certification Authority
Entrust.net Certification Authority (2048)
Entrust.net Secure Server Certification Authority
VeriSign Class 3 Public Primary Certification Authority
VeriSign Class 2 Public Primary Certification Authority
VeriSign Class 1 Public Primary Certification Authority
VeriSign Class 4 Public Primary Certification Authority - G2
VeriSign Class 3 Public Primary Certification Authority - G2
VeriSign Class 2 Public Primary Certification Authority - G2
VeriSign Class 1 Public Primary Certification Authority - G2
VeriSign Class 4 Public Primary Certification Authority - G3
VeriSign Class 3 Public Primary Certification Authority - G3
VeriSign Class 2 Public Primary Certification Authority - G3
VeriSign Class 1 Public Primary Certification Authority - G3
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
RSA Secure Server Certification Authority

There is no way of adding just a couple of the CA certificates it is all or none, but you are able to remove any of them at a later stage.

-strong

Check that the password entered satisfies the minimum requirements for the passwords strength. The minimum requirements for a password are as follows:

- The minimum password length is 14 characters.
- A password needs to have at least one lower case character, one upper case character, and one digit or special character (eg. *$#% etc, a space is classified as special characters).
- Each character should not occur more than three times in a password.
- No more than two consecutive characters of the password should be identical.
- All characters mentioned above are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Display Key Database Password Expiry

The expiry key database command simply displays the date that the password associated with the identified key database will expire. When identifying the key database you simply need to specify the file name of the key database.

The syntax for displaying the expiry of the password associated with a key database with GSKCapiCmd is as follows:

gsk*7*capicmd -keydb -expiry -db <name> [-pw <passwd>]

Where:

object -keydb

action -expiry

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘””, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –expiry –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk*7*capicmd for a value (for example a password) quoting

the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-pw <passwd>
The password for the key database identified by the –db tag above. The –pw tag is required if the key database was created with a password. If the key database does not have a password the –pw tag is not required.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

IMPORTANT: An expiry of 0 means that the password associated with the key database does not expire.

Chapter 3. Certificate Commands

The certificate commands are associated with the -cert object. This object supports the following actions:

- Create a self signed certificate in a key store (-create)
- Add a certificate to a key store (-add)
- Delete a certificate from a key store (-delete)
- Display the details of a certificate in a key store (-details)
- Export a certificate from a key store to another key store (-export)
- Receive a certificate into a key store (-receive)
- Import a certificate from a key store to another key store (-import)
- Extract a certificate from a key store (-extract)
- List the details of the default certificate in a key store (-getdefault)
- Set the default certificate in a key store (-setdefault)
- List the certificates stored in a key store (-list)
- Modify a certificate in a key store (-modify)
- Sign a certificate (-sign)

Each of the following sections goes into detail on how to use and what options are available for each of the identified certificate actions.

Create a Self-Signed Certificate

A self-signed certificate provides a certificate that can be used for testing while waiting for the officially signed certificate to be returned from the CA. Both a private and public key are created during this process.

The create self-signed certificate command creates a self-signed X509 certificate in the identified key database. A self-signed certificate has the same issuer name as its subject name.

The syntax for creating a certificate in an existing key database with GSKCapiCmd is as follows:

gsk*7*capicmd -cert -create {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> -dn <dist name> [-size <key size>] [-x509version <1 | 2 | **3**>] [-default_cert <yes | **no**>] [-expire <days>] [-secondaryDB <filename> -secondaryDBpw <password>] [-ca <true | false>] [-fips] [-sigalg<md5 | **sha1** | sha224 | sha256 | sha384 | sha512>]

Where:

object -cert

action -create

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>

Fully qualified path name of a key database to store the self-signed certificate.

-crypto <module name>

Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>

The PKCS11 cryptographic device token label.

-pw <passwd>

The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>

Label attached to the certificate. The label is used to uniquely identify the certificate by a human user.

-dn <distinct name>

The X.500 distinguished name that will uniquely identify the certificate. The input needs to be a quoted string of the following format (only CN is required):

CN=common name
O=organization
OU=organization unit
L=location
ST=state, province
C=country
DC=domain component
EMAIL=email address

For Example: “CN=weblinux.Raleigh.ibm.com,O=ibm,OU=IBM HTTP Server,L=RTP,ST=NC,C=US”

Multiple OU are now supported. Simply add additional OU key\value pairs to the specified distinguished name. If the OU value requires a comma (‘,’) it will need to escape it with ‘\\’.

For Example: “CN=weblinux.Raleigh.ibm.com,O=ibm,OU=IBM HTTP Server,OU=GSKit\\, Gold Coast,L=RTP,ST=NC,C=US”

-size <key size>

The size of the new key pair to be created. Key size from 512 to 4096, default 1024.

-x509version <1 | 2 | 3>

The version of X.509 certificate to create, default is 3.

-default_cert <yes | no>

Sets the newly created certificate as the default certificate for the key database. By default the newly created self-signed certificate is not set as the default (no). A default certificate in a key database is used when a specific certificate is not specified during the use of the key database.

-expire <days[1]>

Expiration time of the certificate in days, default 365 days. The duration is 1 to 7300 days (20 years)

-secondaryDB <filename>

A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>

Password for the secondary CMS key database supporting the PKCS11 device.

-ca <true | false>

This tag adds the Basic Constraint extension to the self-signed certificate. The Basic Constraint extension value will be set to true or false depending on what value is associated with the tag.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

-sigalg<md5 | **sha1** | sha224 | sha256 | sha384 | sha512>

The hashing algorithm to be used during the creation of the self signed certificate. This hashing algorithm is used to create the signature associated with the newly created self-signed certificate.

[1] To avoid possible timezone issues the actual valid-from time for the certificate will be set one day in the past.

Add a Certificate

The add certificate command stores a CA certificate into the identified key database. The CA certificate is received as a file with the data encoded as either Base64 (ascii) or binary. It is important to identify the correct format of the file otherwise the operation will fail.

The syntax for adding a certificate in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -add {-db <name> | -crypto <module name>
-tokenlabel <token label>} [-pw <passwd>] -label <label> -file <name> [-format
<**ascii** | binary>] [-trust <**enable** | disable>] [-secondaryDB <filename>
-secondaryDBpw <password>] [-fips]

Where:

object -cert

action -add

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate.

-file <name>

Filename of the certificate to add. If it is ".p7", ".smime" or ".eml" then it is assumed to be a pkcs7 encoding. The first cert will take the 'label' given, all other certs, if present, will be labelled with their subject name.

-format <**ascii** | binary>

Format of a certificate The default is Base64 encoded ascii, additional information about base64 encoding can be found in rfc2045 and rfc3548. The binary format is a binary dump of the DER encoded certificate structure. For additional information refer to ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002

-trust <enable | disable>

Trust status of a CA certificate, where the default is 'enable'. When a CA's certificate trust status is enabled then that CA certificate is permitted to be involved in a certificate chain validation. If the CA's certificate trust status is disabled then it cannot be used to validate any certificates. For example if certificate "ABC" is signed by the CA certificate "VeriSign CA" and "VeriSign CA" is not marked as trusted the validation of "ABC" will fail.

-secondaryDB <filename>

A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>

Password for the secondary CMS key database supporting the PKCS11 device.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Delete a Certificate

The delete command simply removes the certificate with the identified label. Once the delete operation is complete, there is no way of recovering the certificate unless you add the certificate back into the key database.

The syntax for deleting a certificate in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -delete {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -cert

action -delete

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate that is to be deleted.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Display Details of a Certificate

The display certificate details command displays the different details associated with the identified certificate. The details displayed include:

- The label of the certificate.
- The size of the key associated with the certificate.
- The X509 version that the certificate was created.
- The serial number for the certificate.
- The issuer and subject distinguished names.
- The certificate's validity period.
- The fingerprint of the certificate.
- The signature of the algorithm used during creation of the certificate.
- An indication of the certificates trust status.

If more details for the certificate are required use the –showOID option. This option displays a more detailed listing of the certificates details.

The syntax for displaying the details for a certificate in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -details [-showOID] {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -cert

action -details

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-showOID
Display a more in depth listing of the certificate.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate that is to be displayed.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Export a Certificate

The export command exports a single certificate specified by its label from either a CMS or PKCS12 key database to another CMS or PKCS12 key database. During this process no key generation occurs. On successful completion the identified certificate will be in both the source and destination key databases.

The syntax to export a certificate from an existing key database to another key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -export -db <name> [-pw <passwd>] -label <label> [-type <**cms** | pkcs12 >] -target <name> [-target_pw <passwd>] [-target_type <**cms** | pkcs12>] [-encryption <**strong** | weak>] [-fips]

Where:

object -cert

action -export

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified filename of a key database containing the certificate to exported from. If the supplied filename has an extension of either “.p12” or .pfx” then it is assumed that it is in PKCS12 format. If it is “.p7”, “.smime” or “.eml” then it is assumed to be a pkcs7 encoding.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate that is to be exported.

-type <**cms** | pkcs12>
The type of the key database to be exported from, default cms.

-target <name>
Destination key database or file that the certificate is to be exported to. If the supplied filename has an extension of either “.p12” or .pfx” then it is assumed that it is in PKCS12 format. If the target keystore does not already exist it will be created.

-target_pw <passwd>
The password of the destination key database or file.

-target_type <**cms** | pkcs12>
The type of the destination key database or file to be exported to, default cms.

-encryption <**strong** | weak>

The strength of encryption used during the export, default strong. This tag is no longer used as the export restrictions in the USA have been eased. This tag is simply added to this command line tool for backward compatibility reasons; it has no effect on the operation, strong is always used.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Receive a Certificate

The receive certificate command stores a certificate received from a CA that was requested to sign a certificate request. The certificate being received can be in either binary or Base64 encoded ascii. Additional information about base64 encoding can be found in rfc2045 and rfc3548. The binary format is a binary dump of the DER encoded certificate structure. For additional information refer to ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002. During the receiving of the certificate, the certificate is matched to its corresponding certificate request. This certificate request is removed from the key database as it is no longer needed.

If the certificate request is required after receiving the certificate, you will need to use the recreate certificate request command that can be found in chapter four of this manual.

The syntax for receiving a certificate to an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -receive -file <name> [-format <**ascii** | binary>] { -db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] [-default_cert <yes | **no**>] [-fips]

Where:

object -cert

action -receive

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however

when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-file <name>
The file name of the certificate that is to be received, this file can be either binary or base64 encoded.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-default_cert <yes | no>
Sets the newly created certificate as the default certificate for the key database. By default the newly created self-signed certificate is not set as the default (no). A default certificate in a key database is used when a specific certificate is not specified during the use of the key database.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Import a Certificate

The import command imports certificates from either a CMS or PKCS12 key database to either a CMS, PCKS12 or PCKS11 key database. During this process no key generation occurs. On successful completion the identified certificates will be in both the source and destination key databases.

The syntax for importing a certificate from an existing key database to another key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -import -db <name> [-pw <passwd>] [-label <label>] [-type <**cms** | pkcs12>] { -target <name> | -crypto <module name> -tokenlabel <token label>} [-secondaryDB <filename> -secondaryDBpw <password>] [-target_pw <passwd>] [-target_type <**cms** | pkcs11 | pkcs12>] [-new_label <label>] [-fips]

Where:

object -cert

action -import

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>

The fully qualified path name of a key database that contains the certificate that is to be imported from. If the supplied filename has an extension of either “.p12” or .pfx” then it is assumed that it is in PKCS12 format. If it is “.p7”, “.smime” or “.eml” then it is assumed to be a pkcs7 encoding.

-pw <passwd>

The password for the key database or PKCS11 cryptographic device identified by either the -db or -crypto tags respectively.

-label <label>

Label attached to the certificate that is to be imported. If the label tag is missing from the command line the operation will transfer all certificates from the source key database to the target key database. If a certificate in the source key database already exists in the target key database that certificate is not imported.

-type <**cms** | pkcs11 | pkcs12>

The type of the key database to be imported from, default cms.

-target <name>

Destination key database to import the certificate into. If the supplied filename has an extension of either “.p12” or .pfx” then it is assumed that it is in PKCS12 format.

-crypto <module name>

Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>

The PKCS11 cryptographic device token label.

Deleted: OR¶
¶
gsk7capicmd -cert -import -file <name> [-pw <passwd>] -label <label> [-type <**cms**>] { -target <name> | -crypto <module name> -tokenlabel <token label>} [-secondaryDB <filename> -secondaryDBpw <password>] [-target_pw <passwd>] [-target_type <**cms** | pkcs11 | pkcs12>] [-new_label <label>] [-fips]¶
¶

Deleted: -file <filename>¶
The fully qualified path name of a PKCS12 file that contains the certificate that is to be imported from. ¶
¶

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-target_pw <passwd>
The password of the destination key database.

-target_type <**cms** | pkcs12>
The type of the destination key database to be imported to, default cms.

-new_label <label>
The label to be used in the destination key database to identify the imported certificate.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

---

Extract a Certificate

The extract certificate command simply extracts the certificate data from the key database and places it into the identified file. If the file does not exist it will be created. If the file already exists an error indicating this will be returned. The data will be saved as either base64 encoding or binary.

The syntax to extract a certificate from an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -extract {-db <name> | -crypto <module name> -tokenlabel <token label>} -pw <passwd> -label <label> -target <name> [-format <**ascii** | binary>] [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -cert

action -extract

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate that is to be extracted.

-target <name>
Destination file that the certificate is to be extracted to.

Deleted: key database or

Deleted: exported

-format <**ascii** | binary>
Format of a certificate. The default is Base64 encoded **ascii**, additional information about base64 encoding can be found in rfc2045 and rfc3548. The binary format is a binary dump of the DER encoded certificate structure. For additional information refer to ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will

be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

List Details of Default Certificate

The list default certificate details simply lists the following details for the default certificate of the identified key database:

- The label of the default certificate.
- The size of the key associated with the default certificate.
- The X509 version that the default certificate was created.
- The serial number for the default certificate.
- The issuer and subject distinguished names.
- The default certificates validity period.
- The fingerprint of the default certificate.
- The signature of the algorithm used during creation of the default certificate.
- An indication of the default certificate's trust status.

The syntax for listing the details for the default certificate in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -getdefault -db <name> [-pw <passwd>] [-fips]

Where:

object -cert

action -getdefault

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-pw <passwd>

The password for the key database identified by the –db tag above.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Set Default Certificate

The set default certificate command sets a certificate to be used as the default certificate for the identified key database. During this command the current default certificate, if there is one, has it's default setting removed. The new certificate is then set as the default certificate. There can only ever be one default certificate in a key database.

The syntax for setting the default certificate in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -setdefault -db <name> [-pw <passwd>] -label <label> [-fips]

Where:

object -cert

action -setdefault

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>

The fully qualified path name of a key database.

-pw <passwd>

The password for the key database identified by the –db tag above.

-label <label>

Label that uniquely identifies the certificate that is to be set as the default certificate in the identified key database.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

List Certificates

The list certificate command lists all of the certificates store within the identified key database. As an example the following certificate list is displayed for a new key database created with the GSKCapiCmd program. The command used to create this list is as follows: gsk7capicmd –cert –list –db <*database name*> [-pw <*password*>].

Certificates found:
* default, - has private key, ! trusted
Entrust.net Global Secure Server Certification Authority
Entrust.net Global Client Certification Authority
Entrust.net Client Certification Authority
Entrust.net Certification Authority (2048)
Entrust.net Secure Server Certification Authority
VeriSign Class 3 Public Primary Certification Authority
VeriSign Class 2 Public Primary Certification Authority
VeriSign Class 1 Public Primary Certification Authority
VeriSign Class 4 Public Primary Certification Authority - G2
VeriSign Class 3 Public Primary Certification Authority - G2
VeriSign Class 2 Public Primary Certification Authority - G2
VeriSign Class 1 Public Primary Certification Authority - G2
VeriSign Class 4 Public Primary Certification Authority - G3
VeriSign Class 3 Public Primary Certification Authority - G3
VeriSign Class 2 Public Primary Certification Authority - G3
VeriSign Class 1 Public Primary Certification Authority - G3
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
RSA Secure Server Certification Authority

The default key is marked with the '*' symbol and all trusted self-signed (root) certs are listed with a '!' symbol. The '-' symbol is used to show where a private key is present.

The syntax to list the certificates in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -list [<**all** | personal | CA | site>] [-expiry [<number of days>] {-db <name> | -crypto <module name> -tokenlabel <token label>}[-pw <passwd>] [-type <**cms** | pkcs11>] [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -cert

action -list

The list command has four special tags that can be associated with it. These tags are used to identify what type of certificates you are requesting to be displayed. The tags are not required, by default all certificate stored within the key database will be displayed. The below lists all of the tags that can be associated with the list command:

all - List the labels of all certificates in the identified key database, this is the default for the list command.

personal - List all personal certificates in the identified key database.

CA – List all of the certificate authority (CA) certificates in the identified key Database.

site – This tag will return error 228 as it is only added for backward compatibility and potential future enhancements. It is not recommended to use this tag.

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-expiry <number of days>

The expiry tag identifies the number of days from today that a certificate is valid for. If the certificates validity falls within this time it is displayed. To list certificates that have already expired, enter the value 0. If you do not specify this tag it is not applied during the execution of the command.

-db <filename>

The fully qualified path name of a key database. If the supplied filename has an extension of either “.p12” or .pfx” then it is assumed that it is in PKCS12 format. If it is “.p7”, “.smime” or “.eml” then it is assumed to be a pkcs7 encoding.

-crypto <module name>

Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-type <cms | pkcs11>
The type of the key database to be exported from, default cms.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Modify Certificate

The modify certificate command allows a CA's certificate trust status to be enabled or disabled. When a CA's certificate trust status is enabled then that CA certificate is permitted to be involved in a certificate chain validation. If the CA's certificate trust status is disabled then it cannot be used to validate any certificates. For example if certificate "ABC" is signed by the CA certificate "VeriSign CA" and "VeriSign CA" is not marked as trusted the validation of "ABC" will fail. You are able to have any number of trusted CA certificates in the single key database.

The syntax for modifying the trust status of a certificate in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -cert -modify -db <name> [-pw <passwd>] -label <label> -trust <**enable** | disable> [-fips]

Where:

object -cert

action -modify

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘””, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-pw <passwd>
The password for the key database identified by the –db tag above.

-label <label>
Label attached to the certificate.

-trust <enable | disable>
Trust status of a CA certificate, default is enabled. Trust status of a CA certificate, default is enable. When a CA’s certificate trust status is enabled then that CA certificate is permitted to be involved in a certificate chain validation. If the CA’s certificate trust status is disabled then it cannot be used to validate any certificates. For example if certificate “ABC” is signed by the CA certificate “VeriSign CA” and “VeriSign CA” is not marked as trusted the validation of “ABC” will fail.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Sign a Certificate

The sign certificate command allows the signing of a certificate request by an existing certificate stored within a key database. The command accepts a certificate request in its file format and a label of a certificate stored within the key database that contains the private key to be used during the signing process. If a certificate is not identified, the private key of the default certificate in the key database is used during the signing process.

The syntax for signing a certificate with GSKCapiCmd is as follows:

gsk7capicmd -cert -sign {-db <name> | -crypto <module name> -tokenlabel <label>} [-pw <passwd>] -label <label> -target <name> [-format <**ascii** | binary>] [-expire <number of days>] -file <name> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips] [-sigalg<md5 | **sha1** | sha224 | sha256 | sha384 | sha512>] [-sernum <serial number>]

Where:

object -cert

action -sign

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database..

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db tag above.

-label <label>
Label of the certificate that has the private key that should be used for the signing operation.

-target <name>
The name of the file that will contain the signed certificate.

-format <**acsii** | binary>
The format of the signed certificate. The default is Base64 encoded **ascii**, additional information about base64 encoding can be found in rfc2045 and rfc3548. The binary format is a binary dump of the DER encoded certificate structure. For additional information refer to ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002

-expire <number of days[2]>

The expiry tag identifies the number of days from today that a certificate is valid for, default is 365 days.

-file <name>

The name and location of the certificate request to be signed.

-secondaryDB <filename>

A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>

Password for the secondary CMS key database supporting the PKCS11 device.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

-sigalg<md5 | **sha1** | sha224 | sha256 | sha384 | sha512>

The hashing algorithm to be used during the signing of the certificate. This hashing algorithm is used to create the signature associated with the newly created signed certificate.

-sernum

The serial number in conjunction with the issuers name uniquely identifies a certificate. A serial number is normally assigned to a certificate by the certificate authority (CA) that signed the certificate request. This tag has been included to allow the emulation of this process. The -sernum tag accepts two types of values:

1.) Hexidecimal - A hexadecimal value can be passed as the -sernum tags value by perpending a “0x” to the front of the serial number.
2.) String - A string representation of the serial number. The string representation of the serial number is normally displayed in ACSII format.

If the –sernum tag is not passed a random serial number is assigned to the signed certificate.

[2] To avoid possible timezone issues the actual valid-from time for the certificate will be set one day in the past.

Chapter 4. Certificate Request Commands

The certificate request commands are associated with the -certreq object. This object supports the following actions:

- Create a certificate request in an existing key database (-create)
- Delete a certificate request from an existing key database (-delete)
- List the details of a certificate request in an existing key database (-details)
- Extract a certificate request from an existing key database (-extract)
- List the certificates requests stored in an existing key database (-list)
- Re-create the certificate request stored in an existing key database (-recreate)

Each of the following sections goes into detail on how to use and what options are available for each of the identified certificate request actions.

Create a Certificate Request

The create certificate request command creates a new RSA private-public key pair and a PKCS10 certificate request in the specified key database. For CMS key databases, the certificate request information is stored in the file with the ".rdb" extension that is associated with the key database. During the creation process the certificate request is also extract to a file that can be used to send the certificate request to a CA for signing.

The syntax for creating a certificate request in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -certreq -create {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> -dn <dist name> [-size <key size>] -file <name> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips] [-sigalg<md5 | **sha1** | sha224 | sha256 | sha384 | sha512>]

Where:

object -certreq

action -create

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (""). You will also need to escape, using a '\' character, the following characters if they appear in the string values: '!', '\', '"', '`'. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db "/tmp/key.kdb" –pw "j\!jj"). Note however when prompted by gsk7capicmd for a value (for example a password) quoting

the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label to be attached to the certificate request on creation. The user uses this label to uniquely identify the certificate request.

-dn <distinct name>
The X.500 distinguished name that will uniquely identify the certificate. The input needed to be a quoted string of the following format (only CN is required):

CN=common name
O=organization
OU=organization unit
L=location
ST=state, province
C=country
DC=domain component
EMAIL=email address

For Example: “CN=weblinux.Raleigh.ibm.com,O=ibm,OU=IBM HTTP Server,L=RTP,ST=NC,C=US”

Multiple OU are now supported. Simply add additional OU key\value pairs to the specified distinguished name. If the OU value requires a comma (‘,’) it will need to escape it with ‘\\’.

For Example: “CN=weblinux.Raleigh.ibm.com,O=ibm,OU=IBM HTTP Server,OU=GSKit\\, Gold Coast,L=RTP,ST=NC,C=US”

-size <key size>
The size of the new key pair to be created. Key size from 512 to 4096, default 1024.

-file <name>
The file name that the certificate request will be extracted to during the certificate request creation process.

-secondaryDB <filename>

A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>

Password for the secondary CMS key database supporting the PKCS11 device.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

-sigalg<md5 | **sha1** | sha224 | sha256 | sha384 | sha512>

The hashing algorithm to be used during the creation of the certificate request. This hashing algorithm is used to create the signature associated with the newly created certificate request.

Delete Certificate Request

The delete certificate request removes the certificate request from the identified key database. This means that the entry in the ".rdb" associated with the certificate request is deleted.

The syntax for deleting a certificate request in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -certreq -delete {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -certreq

action -delete

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (""). You will also need to escape, using a '\' character, the following characters if they appear in the string values: '!', '\', '"', '`'. This will prevent some command line shells from interpreting specific characters within these values. (e.g.

gsk7capicmd –keydb –create –db "/tmp/key.kdb" –pw "j\!jj"). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate request that is to be deleted.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

List Certificate Request Details

The list certificate request details command simple lists the identified certificate requests details. These details include:

- The label of the certificate request.
- The size of the key associated with the certificate request.
- The subject distinguished name.
- The fingerprint of the certificate.
- The signature of the algorithm used during creation of the certificate.

For a more detailed listing of the certificate requests details please us the -showOID option in the command.

The syntax for listing a certificate requests details in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -certreq -details [-showOID] {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd] -label <label> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -certreq

action -details

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-showOID
Display a more in depth listing of the certificate.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate request that is to be displayed.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to

store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Extract Certificate Request

The extract certificate request command extracts an existing certificate request stored in the specified key database to the identified file in base64 format. The certificate request will still exist within the key database so you are able to extract it as many times as needed. The file that is extracted is the file that is dispatched to a CA for signing.

The syntax for extracting a certificate request from an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -certreq -extract {-db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> -target <name> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -certreq

action -extract

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db or –tokenlabel tags above.

-label <label>
Label attached to the certificate request that is to be extracted.

-target <name>
Destination file that the certificate request is to be extracted to.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

List all Certificate Requests

The list certificate request command lists all of the certificate request labels stored within the identified key database.

The syntax for listing the certificate requests stored within an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -certreq -list { -db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -certreq

action -list

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘””, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db tag or the -crypto tag above.

-secondaryDB <filename>
A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>
Password for the secondary CMS key database supporting the PKCS11 device.

-fips
This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Re-create Certificates Requests

The re-create certificate request command recreates a certificate request from an existing certificate stored within the specified key database. The recreation of a certificate may be required to allow a certificate to be signed by another CA in case there was a problem with the CA that originally signed it.

The syntax to recreate a certificate request in an existing key database with GSKCapiCmd is as follows:

gsk7capicmd -certreq -recreate { -db <name> | -crypto <module name> -tokenlabel <token label>} [-pw <passwd>] -label <label> -target <name> [-secondaryDB <filename> -secondaryDBpw <password>] [-fips]

Where:

object -certreq

action -recreate

options

IMPORTANT: On Unix type operating systems it is recommended to always encapsulate string values associated with all tags in double quotes (“”). You will also need to escape, using a ‘\’ character, the following characters if they appear in the string values: ‘!’, ‘\’, ‘”’, ‘`’. This will prevent some command line shells from interpreting specific characters within these values. (e.g. gsk7capicmd –keydb –create –db “/tmp/key.kdb” –pw “j\!jj”). Note however when prompted by gsk7capicmd for a value (for example a password) quoting the string and adding the escape characters should not be done. This is because the shell is no longer influencing this input.

-db <filename>
The fully qualified path name of a key database.

-crypto <module name>
Indicates a PKCS11 cryptographic device operation, where <module name> is the path to the module to manage the crypto device.

-tokenlabel <token label>
The PKCS11 cryptographic device token label.

-pw <passwd>
The password for the key database identified by the –db tag or the -crypto tag above.

-label <label>
Label attached to the certificate request that is to be recreated.

-target <name>
Destination file that the certificate request is to be recreated to.

-secondaryDB <filename>

A CMS key database used to support the PKCS11 device. A PKCS11 device does not normally have a large amount of space available to store a lot of signer certificates. The signer certificates are used for the validation of certificates when they are added to the PCKS11 device.

-secondaryDBpw <password>

Password for the secondary CMS key database supporting the PKCS11 device.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Chapter 5. Random Password Generation

The GSKCapiCmd program provides its users with the ability to generate random passwords. Users are able to specify the password length and if the generated password is required to conform to GSKit's minimum password requirements.

The random commands are associated with the -random object. This object supports the following actions:

- Create a random password of a specified length. (-create)

The following section goes into detail on how to use and what options are available for each of the identified random actions.

Create a Random Password

The create random password command creates a random string of characters that can be used with other GSKCapiCmd commands that require a password. This command should be used if the user is looking for a truly random password.

The syntax for creating a random password with GSKCapiCmd is as follows:

gsk7capicmd -random -create -length <password length> -strong -fips

Where:

object -random

action -create

options

-length <password length>
The length of the random password. There is a maximum length when the -strong tag is used for this command. The maximum length is 125 character.

-strong
Check that the password entered satisfies the minimum requirements for the passwords strength. The minimum requirements for a password are as follows:

- The minimum password length is 14 characters.
- A password needs to have at least one lower case character, one upper case character, and one digit or special character (eg. *$#% etc, a space is classified as special characters).
- Each character should not occur more than three times in a password.
- No more than two consecutive characters of the password should be identical.

- All characters mentioned above are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

-fips

This disables the use of the BSafe cryptographic library. Only the ICC component which must be successfully intialized in FIPS* mode will be used. If the ICC component does not intialize in FIPS mode then the gsk7capicmd operation will fail.

* When in FIPS mode the ICC component uses algorithms that have been FIPS 140-2 validated

Chapter 6. Help Commands

GSKCapiCmd has an extensive help command system. You are able to get help on what objects are available, what actions are associated with a particular object, and help on how to use each of the actions.

The help commands are associated with the -help object. The syntax for the help commands is as follows:

gsk7capicmd -help <*object*> <*action*>

Where:

<*object*> The object you want to find out information on.

<*action*> The action you are wanting to find out information on. This action must be associated with the identified object. If it is not the system will display the help associated with the requested object.

Examples:

Listing all of the objects and their associated actions.

gsk7capicmd -help

Listing the actions for the -keybd object:

gsk7capicmd -help -keydb

Listing the specific help for the -create action associated with the -keydb object.

gsk7capicmd -help -keydb -create

To find out the different objects and their associated actions please refer to chapters 2, 3 and 4 of this manual.

Chapter 7. Version Commands

The version command simply displays version information associated with the currently installed GSKCapiCmd program.

The version command is associated with the -version object. The syntax is as follows:

gsk7capicmd -version

The version command has no associated actions or objects.

Chapter 8. Error Codes and Messages

GSKCapiCmd, returns GSKKM_OK (0) on success or a positive number indicting the error that has occurred. The following table lists all of the error codes and their associated error messages.

Error Code	Error Message
1	Unknown error occurred
2	An asn.1 encoding/decoding error occurred.
3	An error occurred while initializing asn.1 encoder/decoder.
4	An asn.1 encoding/decoding error occurred because of an out-of-range index or non-existent optional field.
5	A database error occurred.
6	An error occurred while opening the database file, check for file existence and permission.
7	An error occurred while re-opening the database file.
8	Database creation failed.
9	The database already exists.
10	An error occurred while deleting the database file.
11	The database could not be opened.
12	An error occurred while reading the database file.
13	An error occurred while writing data to the database file.
14	A database validation error occurred.
15	An invalid database version was encountered.
16	An invalid database password was encountered.
17	An invalid database file type was encountered.
18	The specified database has been corrupted.
19	An invalid password was provided or the key database has been tampered or corrupted.
20	A database key entry integrity error occurred.
21	A duplicate certificate already exists in the database.
22	A duplicate key already exists in the database (Record ID).
23	A certificate with the same label already existed in the key database.
24	A duplicate key already exists in the database (Signature).
25	A duplicate key already exists in the database (Unsigned Certificate).
26	A duplicate key already exists in the database (Issuer and Serial Number).
27	A duplicate key already exists in the database (Subject Public Key Info).
28	A duplicate key already exists in the database (Unsigned CRL).
29	The label has been used in the database.
30	A password encryption error occurred.

31	A LDAP related error occurred. (LDAP is not supported by this program)
32	A cryptographic error occurred.
33	An encryption/decryption error occurred.
34	An invalid cryptographic algorithm was found.
35	An error occurred while signing data.
36	An error occurred while verifying data.
37	An error occurred while computing digest of data.
38	An invalid cryptographic parameter was found.
39	An unsupported cryptographic algorithm was encountered.
40	The specified input size is greater than the supported modulus size.
41	An unsupported modulus size was found.
42	A database validation error occurred.
43	Key entry validation failed.
44	A duplicate extension field exists.
45	The version of the key is wrong.
46	A required extension field does not exist.
47	The validity period does not include today or does not fall within its issuer's validity period.
48	The validity period does not include today or does not fall within its issuer's validity period.
49	An error occurred while validating validity private key usage extension.
50	The issuer of the key was not found.
51	A required certificate extension is missing.
52	An invalid basic constraint extension was found.
53	The key signature validation failed.
54	The root key of the key is not trusted.
55	The key has been revoked.
56	An error occurred while validating authority key identifier extension.
57	An error occurred while validating private key usage extension.
58	An error occurred while validating subject alternative name extension.
59	An error occurred while validating issuer alternative name extension.
60	An error occurred while validating key usage extension.
61	An unknown critical extension was found.
62	An error occurred while validating key pair entries.
63	An error occurred while validating CRL.
64	A mutex error occurred.
65	An invalid parameter was found.
66	A null parameter or memory allocation error was encountered.
67	Number or size is too large or too small.
68	The old password is invalid.

69	The new password is invalid.
70	The password has expired.
71	A thread related error occurred.
72	An error occurred while creating threads.
73	An error occurred while a thread was waiting to exit.
74	An I/O error occurred.
75	An error occurred while loading CMS.
76	A cryptography hardware related error occurred.
77	The library initialization routine was not successfully called.
78	The internal database handle table is corrupted.
79	A memory allocation error occurred.
80	An unrecognized option was found.
81	An error occurred while getting time information.
82	Mutex creation error occurred.
83	An error occurred while opening message catalog.
84	An error occurred while opening error message catalog.
85	An null file name was found.
86	An error occurred while opening files, check for file existence and permissions.
87	An error occurred while opening files to read.
88	An error occurred while opening files to write.
89	There is no such file.
90	The file cannot be opened because of its permission setting.
91	An error occurred while writing data to files.
92	An error occurred while deleting files.
93	Invalid Base64-encoded data was found.
94	An invalid Base64 message type was found.
95	An error occurred while encoding data with Base64 encoding rule.
96	An error occurred while decoding Base64-encoded data.
97	An error occurred while getting a distinguished name tag.
98	The required common name field is empty.
99	The required country or region name field is empty.
100	An invalid database handle was found.
101	The key database does not exist.
102	The request key pair database does not exist.
103	The password file does not exist.
104	The new password is identical to the old one.
105	No key was found in the key database.
106	No request key was found.
107	No trusted CA was found
108	No request key was found for the certificate.
109	There is no private key in the key database

110	There is no default key in the key database.
111	There is no private key in the key record.
112	There is no certificate in the key record.
113	There is no CRL entry.
114	An invalid key database file name was found.
115	An unrecognized private key type was found.
116	An invalid distinguished name input was found.
117	No key entry was found that has the specified key label.
118	The key label list has been corrupted.
119	The input data is not valid PKCS12 data.
120	The password is invalid or the PKCS12 data has been corrupted or been created with later version of PKCS12.
121	An unrecognized key export type was found.
122	An unsupported password-based encryption algorithm was found.
123	An error occurred while converting the key ring file to a CMS key database.
124	An error occurred while converting the CMS key database to a key ring file.
125	An error occurred while creating a certificate for the certificate request.
126	A complete issuer chain cannot be built.
127	Invalid WEBDB data was found.
128	There is no data to be written to the key ring file.
129	The number of days that you entered extends beyond the permitted validity period.
130	The password is too short; it must consist of at least {0} characters.
131	A password must contain at least one numeric digit.
132	All characters in the password are either alphabetic or numeric characters.
133	An unrecognized or unsupported signature algorithm was specified.
134	An invalid database type was encountered.
135	The specified secondary key database is in use by another PKCS#11 device.
136	No secondary key database was specified.
137	The label does not exist on the PKCS#11 device.
138	Password required to access the PKCS#11 device.
139	Password not required to access the PKCS#11 device.
140	Unable to load the cryptographic library.
141	PKCS#11 is not supported for this operation.
142	An operation on a PKCS#11 device has failed.
143	The LDAP user is not a valid user. (LDAP is not supported by this program)
144	The LDAP user is not a valid user. (LDAP is not supported by this

	program)
145	The LDAP query failed. (LDAP is not supported by this program)
146	An invalid certificate chain was found.
147	The root certificate is not trusted.
148	A revoked certificate was encountered.
149	A cryptographic object function failed.
150	There is no certificate revocation list data source available.
151	There is no cryptographic token available.
152	FIPS mode is not available.
153	There is a conflict with the FIPS mode settings.
154	The password entered does not meet the minimum required strength.
200	There was a failure during initialization of the program.
201	Tokenization of the arguments passed to the GSKCapiCmd Program failed.
202	The object identified in the command is not a recognised object.
203	The action passed is not a known -keydb action.
204	The action passed is not a known -cert action.
205	The action passed is not a known -certreq action.
206	There is a tag missing for the requested command.
207	The value passed with the –version tag is not a recognised value.
208	The value passed with the –size tag is nota recognised value.
209	The value passed in with the –dn tag is not in the correct format.
210	The value passed in with the –format tag is not a recognised value.
211	There was an error associated with opening the file.
212	PKCS12 is not supported at this stage.
213	The cryptographic token you are trying to change the password for is not password protected.
214	PKCS12 is not supported at this stage.
215	The password entered does not meet the minimum required strength.
216	FIPS mode is not available.
217	The number of days you have entered as the expiry date is out of the allowed range.
218	Password strength failed the minimum requirements.
219	No Default certificate was found in the requested key database.
220	An invalid trust status was encountered.
221	An unsupported signature algorithm was encountered. At this stage only MD5 and SHA1 are supported.
222	PCKS11 not supported for that particular operation.
223	The action passed is not a known –random action.
224	A length than less than zero is not allowed.
225	When using the –strong tag the minimum length password is 14 characters.
226	When using the –strong tag the maximum length password is 300

	characters.
227	The MD5 algorithm is not supported when in FIPS mode.
228	The site tag is not supported for the –cert –list command. This attribute is simply added for backward compatibility and potential future enhancement.
229	The value associated with the -ca tag is not recognised. The value must be either ‘true’ or ‘false’.
230	The value passed in with the –type tag is not valid.
231	The value passed in with the –expire tag is below the allowed range.
232	The encryption algorithm used or requested is not supported.
233	The target already exists.

Appendix 1. CMS Key Database

What is a CMS Key Database?

CMS is the native GSKit key database (keystore) containing X.509 certificates, certificate requests (ones pending signing by an authority), and private keys for those certificates should they have them. Typically only personal certificates contain private keys. Should a certificate have an associated private key it is stored encrypted in the keystore with its associated certificate. Private keys cannot be stored without an associated certificate.

How is one organized?

A cms keystore consists of a file with extension .kdb and possibly two other files with extension .rdb and .crl respectively.

A key record in a .kdb file is either a certificate or a certificate along with its encrypted private key information. Private keys cannot be stored in a CMS keystore without a corresponding certificate.

When a certificate request is created, a .rdb file with the same file stem as the key database file is created to store the requested key pair along with the PKCS#10 certificate request data. Only when a signed certificate is obtained from a signing authority and received into the key database (the signed certificate is matched up with the private key in the .rdb file and together they are added to the .kdb file as a certificate and its private key information), the request entry is deleted from the request key database.

A .crl file is created for legacy reasons (in the past it contained CRLs) and is no longer used. This file is always empty.

How is it protected?

GSKit implements password protection for access control, confidentiality, and integrity of the CMS key database; the password must be provided prior to any access to the keystore database.

The access control does not limit unauthorized users from reading and writing the file, GSKit relies on the OS for these protections, but because all sensitive data in the keystore is encrypted, all records hashed and the index to all records hashed, access to sensitive data is effectively controlled and any modification to the file is detectable. Should tampering be detected GSKit will not allow access to the keystore similarly as if the incorrect password had been given.

What can I put in a cms key database?

The CMS keystore contains X.509 certificates along with their private key information should they have any. For example when using the GSKCapiCmd tool to

maintain a cms keystore the following items may typically end up in the cms keystore:

CA Certificates. Each valid X.509 certificate needs to be signed. Typically this is done by a trusted Certificate Authority (CA). In order to validate a certificate signed by a CA that CA's public certificate must be in the cms keystore. The GSKCapiCmd tool automatically populates a new cms keystore with a number of CA certificates. Should the cms keystore not already contain a required CA certificate an administrator would typically use GSKCapiCmd to add it . Any valid X.509 certificate can be imported into a cms keystore. In order for the import or add operation to succeed the incoming certificate will be validated. For this reason the certificate needed for the validation chain must be in the keystore already.

Intermediate Certificates. A valid certificate does not necessarily need to be signed by a CA. Instead it can be signed by what is known as an intermediate certificate that has itself be signed by a CA. If this is the case then both the CA certificate and the intermediate certificate must be in the cms keystore in order to validate that certificate. This is known as a certificate validation chain. An administrator would typically use GSKCapiCmd to add their intermediate certificate. Intermediate certificates are treated the same as a CA certificate by the keystore.

Personal Certificates: In a client-server relationship the server may ask the client for its certificate. When acting as a client GSKit will attempt to use a personal certificate from the keystore to present to the server. Typically GSKit will pick the certificate that is marked as the default or the one otherwise indicated by the client application. Another use of a personal certificate is for signing other certificates. For example an Intermediate certificate together with its private key may be in a cms keystore and used to sign other certificates. An extracted version (without the private key) of the intermediate certificate would then be added to other cms keystores to be used in a validation chain. An administrator would typically use GSKCapiCmd to add and extract their personal certificate(s). The final use for a personal certificate is for a server application. A personal certificate may be presented to the client by the server duing a SSL handshake.

GSKCapiCmd supports two certificate transfer formats (eg. for add, import, extract, etc). These are refered to as ascii and binary thougout this document. The default is Base64 encoded ascii. Additional information about base64 encoding can be found in rfc2045 and rfc3548. The binary format is a binary dump of the DER encoded certificate structure. For additional information refer to ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002

What is a label?

A label is a friendly name that an administrator can attached to a certificate contained in a CMS keystore. It is simply a convenient human readable way to reference a certificate.

How can I manipulate certificates in a cms keystore?

Certificates in a CMS keystore are standard X.509 certificates. X.509 entities can be imported (for personal certificates), added (for a certificate needed in a validation chain such as a CA or intermediate certificate), exported (from one CMS keystore to another. This takes the private key with it if one exists), and extracted (extract the public certificate – the private key is not extracted). Note that as GSKCapiCmd does not support PKCS#12 the private key component of a certificate cannot be extracted, only exported into another CMS keystore (along with the certificate itself).

An administrator can also change trust status of a certificate. When a certificate's trust status is enabled that certificate is permitted to be involved in a certificate chain validation. If the certificate's trust status is disabled then it cannot be used to validate any other certificates. For example if certificate "ABC" is signed by the CA certificate "VeriSign CA" and "VeriSign CA" is not marked as trusted the validation of "ABC" will fail.

Appendix 2. A Simple Example

The example below offers an example scenario for a company setting up a website for its employees to access business sensitive information. It is assumed that the web server chosen by the company uses GSKit as it's SSL provider. The example does not cover all issues for such a scenario but instead concentrates on what an administrator would typically need to do to set up a cms keystore in such an environment.

The requirement

The ACME company wishes to set up a website for it's employees to access certain sensitive business information across a wide geographical area. Some employees are more senior than others and therefore will be allowed access to more resources on the server than the more junior employees. It is expected that employees can be assured they are connecting to their company website and not some fraudulent site pretending to be their company site. Employees use a customized web browser that can read CMS keystores to access certificates contained in them.

The CEO of ACME has asked the system administrator to implement this system in a manner that is secure and cost conscious.

Considerations for the administrator

The administrator makes some decisions based on the requirements:

- As the employees are located at different geographical locations a secure channel for the web traffic must be used. The administrator decides that this should be SSL.
- As employees will have different levels of access to web content the administrator decides that the server will operate in client authentication mode where each connecting client must present a valid certificate in order to gain access. Information from this presented certificate will be used to limit access to authorized areas of the web server only (this is outside the scope of this scenario)
- As cost is an issue the administrator decides that it is too costly to have every employee certificate signed by a CA. The administrator decides to make use of a company wide intermediate certificate that will then be used to sign all employee certificates.
- Employees must be able to validate the server's certificate thereby proving the authenticity of the web server.
- The administrator notes that it is bad practice to use a certificate for more than one purpose so decides that another certificate must be produced and signed by the CA. This certificate will be the server certificate used for the website. Using the Intermediate Certificate for this purpose would be poor practice.

Step 1 – Obtain a company wide Intermediate Certificate

The administrator needs to create a certificate that can be used to sign each employee's certificate. This intermediate certificate itself may be publically published

so it needs to be signed by a trusted CA. To achieve this, the administrator performs the following actions:

1. The administrator creates a new cms keystore using the “Create a Key Database” command:

   gsk7capicmd -keydb -create -db acme.kdb -pw offs64b -expire 365 -fips

2. The administrator notices that the new keystore, while containing a number of CA certificates, does not contain the certificate of the CA he would like to use to sign his Intermediate Certificate. He obtains the CA certificate (this is usually done by visiting a well know site that publishes these) and adds it to his cms keystore via the “Add a Certificate” command:

   gsk7capicmd -cert -add -db acme.kdb -pw offs64b -label OurCA -file CACert.arm -format ascii -fips

3. The administrator then creates a new certificate request to be sent to the CA that he has chosen to sign our Intermediate Certificate using the “Create a Certificate Request” command:

   gsk7capicmd -certreq -create -db acme.kdb –pw offs64b -label OurIntermediate -dn “CN=acme.com,O=acme,C=US” -file certreq.arm -fips -sigalg sha1

4. The administrator takes the request file (certreq.arm in this case) and sends it to the CA for signing. Sometime later the signed certificate is returned by the CA. The administrator then receives the certificate into the cms keystore using the “Receive a Certificate” command:

   gsk7capicmd -cert -receive -file signedCert.arm -db acme.kdb -pw offs64b -fips

5. Make the new certificate the default one. This means that it will be used by default to sign other certificate request if no other certificate lable is given. The administrator makes it the default certificate using the following command:

   gsk7capicmd -cert -setdefault -db acme.kdb -pw offs64b -label OurIntermediate -fips

Step 2 – Sign all employee certificates using the ACME Intermediate

The administrator now has a cms keystore containing ACME’s intermediate certificate and the CA certificate that signed that intermediate certificate. The administrator now needs to use ACME’s intermediate certificate to sign all the employee certificates. To achieve this, the administrator performs the following actions:

1. The administrator asks each employee to obtain the CA certificate and add it to their respective cms keystores. Note that employees may first need to create their

own cms keystore in the same manner as the administrator did in item 1 of step 1. The employee adds the CA certificate using the “Add a Certificate” command:

gsk7capicmd -cert -add -db Dave.kdb -pw Davepwd -label OurCA -file CACert.arm -format ascii -fips

2. The administrator extracts the Intermediate Certificate (note that this does not extract the private key of the certificate) using the “Extract a Certificate” command:

   gsk7capicmd -cert -extract -db acme.kdb -pw offs64b -label acmeCert -target acmeCert.arm -fips

3. The administrator sends the ACME intermediate certificate to each employee asking them to add it to their keystore. Employees do this via the “Add a Certificate” command:

   gsk7capicmd -cert -add -db Dave.kdb -pw Davepwd -label acmeCert -file acmeCert.arm –format ascii –fips

4. The administrator asks each employee to create a certificate request putting their employee email address in the CN field. The employees use the “Create a Certificate Request” command:

   gsk7capicmd -certreq -create -db Dave.kdb –pw Davepwd -label myCert -dn “CN=dave@acme.com,O=acme,C=US” -file DavesCertReq.arm -fips -sigalg sha1

5. Upon receipt of each employee’s certificate request the administrator signs it and returns the signed certificate to the employee. The administrator uses the “Sign a Certificate” command to achieve this:

   gsk7capicmd -cert -sign -db acme.kdb -pw offs64b -label acmeCert -target DavesCertReq.arm -expire 365 -file DavesSignedCert.arm -fips -sigalg sha1

6. As each employee obtains their signed certificate they receive it into their cms keystore. Employees use the “Receive a Certificate” command:

   gsk7capicmd -cert -receive -file DavesSignedCert.arm -db Dave.kdb -pw Davepwd -fips

7. Make the new certificate the default one. This means that it will be the certificate sent to the web server when it requests one via SSL for client authentication purposes. The employee makes it the default certificate using the following command:

   gsk7capicmd -cert -setdefault -db Dave.kdb -pw Davepwd -label myCert -fips

Step 3. Create the web Server certificate

At this stage the administrator has a cms database containing the CA certificate and the ACME Intermediate certificate (with it’s corresponding private key). The

administrator puts this cms keystore in a safe place using it only to sign new employee certificates.

Each employee has a cms keystore containing the CA certificate, the ACME Intermediate Certificate (minus the corresponding private key), and their own certificate that has been signed by the ACME Intermediate Certificate.

The remaining task for the administrator is to create a cms keystore with a certificate to be used by the web server. Although the administrator could have used the ACME Intermediate Certificate for this purpose as stated previously it is considered bad practice to use a certificate for more than one purpose. The intermediate certificate is already being used to sign employees certificates. To create a keystore and server certificate the administrator performs the following actions:

1. The administrator creates a new cms keystore using the “Create a Key Database” command:

   gsk7capicmd -keydb -create -db acmeWebServer.kdb -pw ejed43dA -expire 365 -fips

2. The administrator adds the CA certificate to the keystore using the “Add a Certificate command:

   gsk7capicmd -cert -add -db acmeWebServer.kdb -pw ejed43dA -label OurCA -file CACert.arm –format ascii -fips

3. The administrator creates a new certificate request to be sent to the CA that he has chosen to sign our web server certificate using the “Create a Certificate Request” command:

   gsk7capicmd -certreq -create -db acmeWebServer.kdb –pw ejed43dA -label OurServerCert -dn “CN=web.acme.com,O=acme,C=US” -file serverCertReq.arm -fips -sigalg sha1

4. The administrator takes the request file (serverCertReq.arm in this case) and sends it to the CA for signing. Sometime later the signed certificate is returned by the CA. The administrator then receives the certificate into the cms keystore using the “Receive a Certificate command:

   gsk7capicmd -cert -receive -file signedServerCert.arm -db acmeWebServer.kdb -pw ejed43dA -fips

5. Make the new certificate the default one. This means that when a client connects to the web server the server will offer this certificate to the client. The administrator makes it the default certificate using the following command:

   gsk7capicmd -cert -setdefault -db acmeWebServer.kdb -pw ejed43dA -label OurServerCert -fips

6. The administrator now has a cms keystore with a server certificate ready for use by the web server application.

So do we meet the requirements?

Let's look at each requirement in turn:

- As the employees are located at different geographical locations a secure channel for the web traffic must be used. The administrator decides that this should be SSL.

  For a web server to make use of SSL it must have a server side certificate to offer to clients during the SSL handshake. The certificate labelled "OurServerCert " in the keystore acmeWebServer.kdb may be used for this purpose.

- As employees will have different levels of access to web content the administrator decides that the server will operate in client authentication mode where each connecting client must present a valid certificate in order to gain access. Information from this presented certificate will be used to limit access to authorized areas of the web server only (this is outside the scope of this scenario)

  Each employee has their own certificate to offer to the server when it requests one during the SSL handshake. The server can first validate the client certificate as it has the signer chain, that is, the client certificate is signed by the ACME Intermediate Certificate, and the ACME intermediate certificate is in turn signed by the CA certificate. The server keystore (acmeWebServer.kdb) has both of these. Once the client certificate has been validated the application can inspect the CN of the certificate and extract the employee email address from it. The application can then use the employee email address to determine the level of access allowed for the connection.

- As cost is an issue the administrator decides that it so too costly to have every employee certificate signed by a CA. The administrator decides to make use of a company wide intermediate certificate that will then be used to sign all employee certificates.

  The administrator only incurred the expense of two CA signing operations. One for the Intermediate Certificate and one for the signer certificate.

- Employees must be able to validate the server's certificate thereby proving the authenticity of the web server.

  When the client application receives (as part of the SSL handshake) the server certificate it can verify the validity of that certificate as it has the CA certificate that signed it.

- The administrator notes that it is bad practice to use a certificate for more than one purpose so decides that another certificate must be produced and signed

by the CA. This certificate will be the server certificate used for the website. Using the Intermediate Certificate for this purpose would be poor practice.

Two certificates have been created. "OurServerCert" for use of the ACME web server and "OurIntermediate" for the administrator to use to sign employee certificates.

Appendix 3. 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 106-0032, Japan

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

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

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

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

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

IBM Corporation
2ZA4/101
11400 Burnet Road
Austin, TX 78758
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.

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

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

Trademarks

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

IBM
IBM logo
AIX
DB2
Novell
SecureWay
Tivoli
Tivoli logo
Universal Database
WebSphere

Lotus is a registered trademark of Lotus Development Corporation and/or IBM Corporation.

Domino is a trademark of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both.

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

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the U.S., other countries, or both.

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

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