Gentran Integration Suite™

Hardware Security Module

Version 4.3

Sterling Commerce
An IBM Company

Store System Certificates on a Hardware Security Module (HSM)

Within Application, a digital certificate for which the private key is maintained is called a *system certificate*. You can store system certificates in a database using Application, or on a *Hardware Security Module* (HSM) — a hardware-based security device that generates, stores, and protects cryptographic keys. Application supports the SafeNet Eracom ProtectServer Orange External and ProtectServer Gold PCI devices.

The SafeNet Eracom architecture divides the HSM into multiple slots. Each slot has an associated security provider and can be protected by a separate Personal Identification Number (PIN). You can create a separate slot on the HSM for Application and protect the slot with a unique PIN. The provider for the default slot 0 is ERACOM. Providers for additional slots are named ERACOM.*n*, where *n* is the number of the slot.

After you store your system certificates in the HSM and import the system certificate information into Application, all system certificates are displayed and available when you configure Application.

Use an HSM with Application

Before you can use an HSM with Application, you must configure Application to use and recognize the SafeNet Eracom HSM.

Configure HSM

To install and set up the SafeNet Eracom HSM, follow the instructions provided by the vendor; ensure that you install Java Runtime. Use the provider for the slot where Application keys will be stored when you set up and use the utilities.

Note: After you create a PIN for the SafeNet Eracom slot, do not change the PIN. Application cannot access a key on the HSM if you change the PIN.

Configure Application

To configure Application to use the SafeNet Eracom HSM:

1. Change to the *install_dir*/bin directory.
2. Add the following lines to the tmp.sh and tmp.sh.in files:

```
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/Eracom/lib
export LD_LIBRARY_PATH
```

3. If you are configuring a network-based server, add the following lines to the tmp.sh and tmp.sh.in files, where *network_device_IP_OR_hostname* is the IP address or fully qualified domain name of the SafeNet Eracom network-based server:

   ```
   ET_HSM_NETCLIENT_SERVERLIST=network_device_IP_OR_hostname
   export ET_HSM_NETCLIENT_SERVERLIST
   ```

4. Copy the jprov.jar from the /opt/Eracom/lib directory to the *install_dir*/jdk/jre/lib/ext directory.
5. Add a definition for each security provider to the *install_dir*/jdk/jre/lib/security/java.security file. To add a definition, identify the number assigned to the Certicom provider and assign n+1 to the SafeNet Eracom provider. For all other providers identified after the SafeNet Eracom provider, increase the security.provider number by 1.

   ```
   security.provider.n=com.certicom.ecc.jcae.Certicom
   security.provider.n+1=au.com.eracom.crypto.provider.ERACOMProvider
   ```

   If you are using a slot other than zero on the SafeNet Eracom HSM, specify the slot as follows, where *x* is the number of the slot:

   ```
   security.provider.n+1=au.com.eracom.crypto.provider.slotx.ERACOMProvider
   ```

6. Modify the *install_dir*/properties/security.properties file.
   - Define the TLSProviderPolicy command:

     If the provider is defined in slot 0, ensure that the only uncommented line for the TLSProviderPolicy parameter is the following:

     ```
     TLSProviderPolicy=
     TLS:*:ECMQV:P:.CT;TLS:SIG:MD2withRSA:P:ERACOM;TLS:Cipher:RawRSA:P:ERACOM
     ;TLS:*:RSA:P:ERACOM;TLS:*:*:P:Certicom
     ```

     If the provider is defined in a slot other than 0, modify the TLSProviderPolicy parameter as follows, where *x* is the slot you are configuring:

     ```
     TLSProviderPolicy=

     TLS:*:ECMQV:P:.CT;TLS:SIG:MD2withRSA:P:ERACOM.x;TLS:Cipher:RawRSA:P:ERAC
     OM.x;TLS:*:RSA:P:ERACOM.x;TLS:*:*:P:Certicom
     ```

   - Define the KeyStoreProviderKey command as follows:

     If the provider is defined in slot 0, ensure that KeyStoreProviderMap is defined as:

     ```
     KeyStoreProviderMap=SCIKS,SCIKS,false,Certicom,Certicom,false;
     nCipher.sworld,nCipherKM,false,nCipherKM,nCipherKM,true;CRYPTOKI,ERACOM,
     true,ERACOM,ERACOM,true
     ```

     If the provider is defined in any slot other than 0, modify the KeyStoreProviderMap parameter as follows, where *x* is the slot number:

     ```
     KeyStoreProviderMap=SCIKS,SCIKS,false,Certicom,Certicom,false;
     nCipher.sworld,nCipherKM,false,nCipherKM,nCipherKM,true;
     CRYPTOKI,ERACOM.x,true,ERACOM.x,ERACOM.x,true
     ```

Manage System Certificate Utilities

To manage system certificates on an HSM, system certificate utilities are provided. The utilities are located in the /*install_dir*/bin directory.

Note: Before using the system certificate utilities, ensure that the Application database is running and Application is not running.

Create System Certificates

Use CreateSystemCert.sh to create a self-signed system certificate to store on the HSM. Following is the syntax:

```
./CreateSystemCert.sh storetype provider autogen totrusttable signingbit keytype
keysize keyname rfc1779rdnsequence serial validityindays [system passphrase]
[store passphrase] [key passphrase]
```

If you do not provide the system passphrase, store passphrase, and key passphrase on the command line, you are prompted for them. The following sample command creates a system certificate to store on an SafeNet Eracom HSM:

```
./CreateSystemCert.sh CRYPTOKI ERACOM false false true RSA 1024 hsmkey "CN=hsmkey" 13
365 password 999999 999999
```

CreateSystemCert.sh Parameters

The following table provides the parameters for the CreateSystemCert.sh script.

Parameter	Description	Valid Values
storetype	The keystore type. CRYPTOKI is the only keystore type supported.	CRYPTOKI
provider	The provider of the keystore type.	ERACOM[.N]
autogen	Whether to use system generated information to control access to the key and keystore. Set to **false** for keys on HSMs.	False
totrusttable	Determines if the certificate is added to the trusted certificate table.	True | false
signingbit	Sets the sign key usage bit for the self-signed certificate.	True | false
keytype	The public key algorithm.	RSA is the only supported algorithm
keysize	The length, in bits, of the RSA modulus.	768, 1024, 2048, 3072, or 4096
keyname	The name of the Application system key to create.	Key name

Parameter	Description	Valid Values
rfc1779rdnsequence	The distinguished name string field contains any of the fields identified in the Valid Values column. Only the CN field is required. Separate each field with a comma. Following is a sample distinguished name definition: `“CN=HSMKey,O=SCI,C=US”`	Valid information: ◆ CN = CommonName ◆ O = Organization ◆ OU = Organization Unit ◆ L = Location ◆ ST = State ◆ C = Country For C, provide a two-letter ISO3166-1 alpha-2 code
serial	The certificate serial number.	A number value
validityindays	How long, in days, the certificate is valid.	Number of days
system passphrase	The Application system passphrase. This value is optional on the command line. If you do not provide it, you are prompted for it.	Application system passphrase
store passphrase	The passphrase for accessing the keystore. This value is optional on the command line. If you do not provide it, you are prompted for it.	The PIN for the token on the SafeNet Eracom HSM where the keystore resides.
key passphrase	The passphrase for the private key. This value is optional on the command line. If you do not provide it, you are prompted for it.	The PIN for the token on the SafeNet Eracom HSM where the keystore resides.

Import System Certificates

Use the Import System Certificates script (ImportSystemCert.sh) to import certificates and keys to the HSM keystore, list system certificates stored on an HSM, and import system certificate information into the Application database.

Import a System Certificate and Key to the HSM

You can import a system certificate to the HSM in keycert, pkcs12, or pem format. Importing a system certificate adds the key and certificate to the HSM and creates a corresponding entry in the Application database. If you import a pem type certificate and key, make sure that the private key is created in DES- or triple-DES encrypted format.

Following is a sample pem private key, created in triple-DES format:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,CE0243B4833BD321

RtN+AFGTmx6EROcbo8fMXnMaRM/JcKIc3jbKYB5t6H6H5uvUrAmv+Si62QEtqg9V
x5r+GhiLcA9sdllKpnIXYg63Y+egn8DsxdGUCqnC+HDUlRVHX0NWKJ3FwXukr9iN
WP4MBr+NXMSETaBA0O0B4oSRCWvxelc2U2GItvUqJsOjLSILbahAgZk/j6LUDMy4
2FWoRtWZyGVz/gc+pN+bOwFHpbrZxd1YqZGRNKeZKTpXWslqxp5NDraBl1cmJ3vL
```

```
0RTnkwZnnyJ1Brc/WynlVfRK1gEEg8MPa3B9veat70ET/mLERuA4Ke8r0WAy5Y/w
7YowicmwbO4q7RLVLm1ZmvPF4OXL8xIvaIUMOCW8/MNpanxZ4BB1CfTwQKQ9koJ7
9MT8K8ofu6V9TSK4Rw1cCpTKvattg/H72Ut39Yz185Ec+E8sV0BtilpqVsYSt1g6
lO805MqPym6gPo2NLpvk1iPLUZ1vIfthz+qb5cyXj1ng9aZSeRF/lytPLxSSy3LN
J9SZrnfHwbuhnyuQmco3SsCtYXnZ81cDHX+4O8sGqHA1zMwuqErorUvwxD6ZNnlc
DTmKIt826oows4Gtw48aEwjV4lk8FXQsWQjDWJHjFNNvGiyszPJjvPvM8zL1EwxO
mJFeNxBb0U3zgLs5aK/HHRn1/gzOBHwtr8bdFFBkpLoVGnbW+mRVxmJOvvPe7Zo+
sJXLEWC8Bm4klV8H6ynx6aQJ8a62HqbjPvShq1VH2I+1iwbyE3DzxY5sHrzZA2rb
dHabk3f0nBUvMegKI9Ye4ktLJf8yIQfsSBSJTEYXHqyx5ptoAEIlIQ==
-----END RSA PRIVATE KEY-----
```

Following is the syntax to import a certificate and key to the HSM, for keycert or pem format. See *ImportSystemCert.sh Parameters* on page 8.

```
./ImportSystemCert.sh -keycert|-pem systempass certname file password keystoretype
keystoreprovider storepass keypass
```

Following is the syntax to import a certificate and key to the HSM, for a certificate and key in pkcs12 format. See *ImportSystemCert.sh Parameters* on page 8.

```
./ImportSystemCert.sh -pkcs12 systempass certname pkcs12file pkcs12storepass
pkcs12keypass keystoretype keystoreprovider storepass keypass
```

List System Certificates Stored on an HSM

Use the following command to list information about system certificates stored on an HSM.

```
./ImportSystemCert.sh -keystore keystoretype keystoreprovider storepass keypass
```

The following sample is a command to list information about system certificates stored on a SafeNet Eracom HSM:

```
./ImportSystemCert.sh -keystore CRYPTOKI ERACOM 999999 999999
```

The following sample is output from the command listed above:

```
Key exists with alias rayado-e5305c3-10d8f4bde7f--7fc1
Certificate Subject Info CN=test, OU=test, O=test, L=test, ST=Alabama, C=US
Certificate Issuer Info CN=Pythagoras, OU=System Verification, O=Sterling, L=Dublin,
ST=OH, C=US, EMAILADDRESS=caussuer@company.com
```

Use the alias information returned by this command to import information about a certificate stored on the HSM to the Application database. In the previous sample, the alias is rayado-e5305c3-10d8f4bde7f--7fc1**.**

Import HSM System Certificate to the Application Database

Use the ImportSystemCert command to import information about a system certificate stored on the HSM to the Application database. Use this procedure when a key and certificate already exist on the HSM and were added to the HSM independent of Application. You must import the information for a system certificate stored on an HSM to the database before it can be used by Application.

Note: Depending upon the method used to add the private key and certificate to the HSM, the list function may display duplicate entries for a single key and certificate pair.

You must obtain the system certificate alias before you can import information about a system certificate to the database.

```
./ImportSystemCert.sh -keystore systempass certname alias keystoretype
keystoreprovider storepass keypass
```

Following is a sample command to import information about the system certificates stored on the HSM.

```
./ImportSystemCert.sh -keystore systempass certname alias CRYPTOKI ERACOM
999999 999999
```

ImportSystemCert.sh Parameters

The following table provides the parameters for the ImportSystemCert.sh script.

Parameter	Description	Valid Values
Certtype	The certificate type to import. Four types of certificate files are supported: pkcs12, pkcs8, pem, and keystore. Application only supports pem keys encrypted with DES or 3DES. Use keystore to list or import the keystore.	keycert, pkcs12, pem, keystore
systempass	The Application system passphrase.	Application system passphrase
certname	The name to assign the certificate in the Application database.	Certificate name
pkcs12file	The pkcs12 file to import.	Name of file
file	Keycert or PEM file to import.	Name of file
password	Store passphrase for the keycert or PEM file.	Valid passphrase
alias	The key name stored in the HSM.	key name on the HSM Only alias names containing characters a-z, A-Z, 0-9 or hyphen (-), and whose total length is no longer than the system GUID length.
keystoretype	The keystore type to import.	CRYPTOKI
keystoreprovider	The provider type. SafeNet Eracom is the only HSM supported.	ERACOM or ERACOM.n if you are importing certificates to a slot other than the default slot 0.
pkcs12storepass	The store passphrase for the PKCS12 file.	Valid passphrase
pkcs12keypass	The key passphrase used to encrypt the private key in the PKCS12 file.	Valid passphrase
storepass	The PIN for the token protecting the SafeNet Eracom HSM where the keystore resides.	Valid PIN for the token

Parameter	Description	Valid Values
keypass	The PIN for the token protecting the SafeNet Eracom HSM where the keystore resides.	Valid PIN

View System Certificates

If you plan to delete system certificates from the database, you must identify the object ID of the certificate to remove. Use the RemoveSystemCert.sh utility with the -l option to view information about certificates in the Application database, including the object ID. The syntax is as follows:

```
./RemoveSystemCert.sh -l
```

You are prompted for the Application system passphrase. The sample output is as follows:

```
PrivateKeyInfo for ID= rayado:1ccce3c:10b625d0227:-7fb5
Name= B2BHttp Userid= System Administrator
CreateDate= 2006-05-23 12:50:27.0 Issuer= C=US, O=Sterling,
CN=B2B Serial= 4 Subject= C=US, O=Sterling, CN=B2B Usage= All NotBefore2006-05-23
NotAfter2008-05-23 Status = 0
```

Remove System Certificates

Use RemoveSystemCert.sh to remove system certificate records from both the Application database and the HSM. After you type the command, you are prompted for the Application system passphrase. The syntax is as follows:

```
./RemoveSystemCert.sh -r [ObjectID]
```

Caution: This procedure permanently deletes the system certificate from the HSM. The private key data it contains cannot be recovered.

The following table provides the parameters used when removing certificates.

Parameter	Description	Valid Values
-r	Removes the record identified in the ObjectID variable. You obtain the object ID by running the RemoveSystemCert.sh -l utility. See *View System Certificates* on page 9.	-r
ObjectID	The ID of the record to remove. See *View System Certificates* on page 9 for instructions on obtaining the object ID.	Certificate ID

Export System Certificates

The ExportSystemCert.sh utility enables you to export system certificates, including the private key, from the Application database. The certificate and key are exported to a PKCS12 keystore. After you type the command, you are prompted for the Application passphrase. The syntax is as follows:

```
./ExportSystemCert.sh keyname pkcs12filename pkcs12storepass pkcs12keypass
```

Note: System certificates on an HSM cannot be exported using ExportSystemCert.sh.

Following is a sample command usage:

```
./ExportSystemCert.sh testkey testkey.txt password password
```

ExportSystemCert.sh Parameters

The following table provides the parameters used when running the ExportSystemCert.sh script.

Parameter	Description	Valid Values
keyname	The name of the Application system key to export.	Valid key name
pkcs12filename	File where exported information is written.	Name of file
pkcs12storepass	Store passphrase that protects the store.	Store passphrase value
pkcs12keypass	Key passphrase that protects the key.	Key passphrase value

About the GenCSR Utility

The GenCSR.sh utility generates a key pair on an HSM and creates a PKCS10 certificate signing request (CSR) with the public key from that key pair. You can then submit the CSR to a Certificate Authority (CA). When you receive a CA-issued certificate, use GenCSR to update the certificate. The system certificate is not available in Application until it is updated with a CA-issued certificate.

You can also use this utility to view a list of CSRs, write information about a CSR to a file, delete a CSR, or write information about a CA-issued certificate stored on the HSM to a file. Information about CSRs is maintained in the Application database, while the actual keys are stored on the HSM.

To use the utility, first determine what action you want to perform. Then, use the GenCSR utility and identify the action in the command line. For each action, supply the arguments required for the action in the properties file. A sample properties file called csr.properties.sample is provided in the /*install_dir*/properties directory. After you type each command, you are prompted for the Application passphrase. The syntax is as follows:

```
./GenCSR.sh -a ACTION -p PROPERTIES
```

GenCSR Utility Arguments

Parameter	Description	Valid Values
-a ACTION	The action to perform.	CREATE, UPDATE, LIST, DELETE, GETPCKS10, GETCACERT

Parameter	Description	Valid Values
-p PROPERTIES	The properties file that contains additional parameters needed for the actions. Identify the path to the file.	Name of a properties file

Generate a PKCS10 Certificate Signing Request (CSR)

The GenCSR.sh program generates a key pair on an HSM and creates a PKCS10 certificate signing request (CSR) with the public key from that key pair. You can then submit the CSR to a Certificate Authority (CA). When you receive a CA-issued certificate, you use GenCSR to update the certificate on the HSM. The system certificate is not available on Application until it is updated with a CA-issued certificate. The syntax is as follows:

```
./GenCSR.sh -a create -p ../properties/csr_create.properties
```

The following table describes the parameters required in the properties file for the create argument.

Parameter	Description	Valid Values
provider	Name of keystore provider.	ERACOM[.N]
keystoretype	Name of the keystore used.	CRYPTOKI
certificate.request.Name	Name of the CSR. This information is written to a file that you send to the CA, to obtain a CA certificate.	Name assigned to the CSR file
algorithm	Public key algorithm.	RSA
key.length	Length, in bits, of the RSA modulus.	512, 768, 1024, 2048
csr.file	Name of the CSR file in which to write information about the CSR, including the path to the file.	File name to write CSR information to
store.password	Passphrase used to protect the keystore.	PIN for the token protecting the SafeNet Eracom HSM where the keystore you are using is located.
key.password	Passphrase used to protect the keystore.	Key passphrase
Distinguished name string	Contains any of the fields identified in the Valid Values column. Only the CN field is required. Following is a sample distinguished name definition: `CN=HSMKey` `O=SCI` `C=US`	Valid information for each category: ◆ CN = CommonName ◆ O = Organization ◆ OU = Organization Unit ◆ L = Location ◆ ST = State ◆ C = Country; must provide a two-digit ISO3166-1 alpha-2 code

Update the HSM Keystore with CA-Issued Certificates

Use the GenCSR utility with the update argument to add CA-issued certificate information to the HSM keystore. The syntax is as follows:

```
./GenCSR.sh -a update -p ../properties/csr_update.properties
```

The following table describes the parameters required in the properties file for the Update Argument.

Parameter	**Description**	**Valid Values**
provider	Name of keystore provider.	ERACOM or ERACOM.n
keystoretype	Name of the keystore used.	CRYPTOKI
certificate.request.Name	Name of the CSR to update.	Name assigned to a CSR
add.trusted	Identifies if the certificate information is added to the trusted certificate table.	True | false
ca.cert.file	Path and file name of the file in which to write information about the CA-issued certificate.	Valid path and file name of a CA-issued certificate file

List Certificate Signing Requests

Use the GenCSR utility with the list argument to display CSRs in the HSM database. The syntax is as follows:

```
./GenCSR.sh -a list
```

Note: No property file is required for the list argument.

Delete a Certificate Signing Request

Use the GenCSR utility with the delete argument to delete a CSR. This utility deletes the CSR only. It does not delete system certificates that are updated with a CA-issued certificate. The syntax is as follows:

```
./GenCSR.sh -a delete -p ../properties/cacert.properties
```

The following table describes the parameters required in the properties file for the Delete Argument.

Parameter	**Description**	**Valid Values**
certificate.request.Name	Name of the CSR to delete.	Name of a CSR
keystoretype	Name of the keystore used.	CRYPTOKI
provider	Name of keystore provider.	ERACOM[.N]

Write CSR Information

Use the GenCSR utility with the getpkcs10 argument to write a CSR in pkcs10 format to the specified file. The syntax is as follows:

```
./GenCSR.sh -a getpkcs10 -p ../properties/csr_getpkcs10.properties
```

The following table describes the parameters required in the properties file for the getpkcs10 Argument.

Parameter	Description	Valid Values
certificate.request.Name	Name of the CSR.	Name assigned to a CSR
keystoretype	Name of the keystore used.	CRYPTOKI
csr.file	Fully qualified path to the file in which to write information about the CSR.	Path and file name of a file to write the CSR information

Write CA-Issued Certificate to a File

Use the GenCSR utility with the getcacert argument to write the certificate issued by the CA to a file. The syntax is as follows:

```
./GenCSR.sh -a getcacert -p ../properties/getcacert.properties
```

The following table describes the parameters required in the properties file for the getcacert Action.

Parameter	Description	Valid Values
certificate.request.Name	Name of the CSR.	Certificate name
keystoretype	Name of the keystore used.	CRYPTOKI
ca.cert.file	Fully qualified path to the file in which to write information about the CA certificate.	Name and path of a CA certificate file

Move System Certificates to the HSM

Use the following procedure to move existing self-signed certificates or CA-issued certificates located in the Application database to an HSM.

Note: It is more secure to regenerate keys and certificates using CreateSystemCert.sh or GenCSR.sh.

To move system certificates from the database to the HSM:

1. Change to the *install_dir*/bin directory.
2. Stop Application and start the database.
3. Export the system certificate to a PKCS12 file; run the following command:

   ```
   ./ExportSystemCert.sh keyname pkcs12filename pkcs12storepass pkcs12keypass
   ```

4. Find the object ID of the system certificate to remove. Type RemoveSystemCert.sh -l.
5. Remove the system certificate from the database:

   ```
   RemoveSystemCert.sh -r xxxx
   ```

   where *xxxx* is the object ID of the certificate you wish to remove

6. To import the system certificate that you exported to the HSM and create a corresponding database entry:

   ```
   ./ImportSystemCert.sh -pkcs12 systempass certname pkcs12file
   pkcs12storepass pkcs12keypass keystoretype keystoreprovider storepass
   keypass
   ```

Caution: If you move the OpsDrv, OpsKey, and UIKey to the HSM, use the exact name. Otherwise, Application will not function properly. For all other system certificates, the name is not critical.

When moving system certificates other than the OpsDrv, OpsKey, and UIKey, the object ID that is used by services and adapters changes. Reconfigure any services that use the system certificates that were moved.

Generate Internal System Certificates on the HSM

There are three system certificates installed with Application to secure internal operations. Little security benefit is provided by moving them to the HSM. Your security policy may require that all certificates containing private keys be stored on the HSM.

When generating the Application internal system certificates called OpsDrv, OpsKey, and UIKey on the HSM, use the exact names. Otherwise, Application will not function properly.

To generate internal system certificates:

1. Type RemoveSystemCert.sh -l to view certificates in the database.
2. Note the object ID for each system certificate.
3. Delete the system certificates from the database by running the following command for each certificate:

   ```
   ./RemoveSystemCert.sh -r xxxx
   ```

   where *xxxx* is the object ID of the certificate you wish to remove.

4. Generate the system certificate on the HSM for each certificate:

   ```
   ./CreateSystemCert.sh storetype provider autogen totrusttable signingbit
   keytype keysize keyname rfc1779rdnsequence serial validityindays [system
   passphrase]
   [store passphrase] [key passphrase]
   ```