December 2009

Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z

M. A. Tebolt
miket@us.ibm.com
IBM Platform Evaluation Test Laboratory
Poughkeepsie, NY

Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z

Table of Contents

Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z

Introduction

This paper describes steps to configure WebSphere® Application Server V7.0 and IBM HTTP Server (IHS) V7.0 to exploit cryptographic hardware to accelerate SSL processing when running on Linux® on IBM System z®. The configurations use clear-key cryptography: The private keys used in the cryptographic operations are stored in a password-protected keystore (a PKCS#12 file in the case of WebSphere), or, stored encrypted in files managed by the PKCS#11 layer (in the case of the HTTP Server). The private keys are extracted and decrypted before being used by the hardware to accelerate the particular cryptographic operation being performed.

There are two types of IBM System z cryptographic hardware used by WebSphere and IHS:

Crypto Express2 feature: Feature code 0863 for IBM eServer™ zSeries® 890 and 990 (z890, z990), IBM System z9® Business Class (z9® BC) and IBM System z9 Enterprise Class (z9 EC), and IBM System z10 Business Class™ and IBM System z10™ Enterpise Class (z10 BC™, z10 EC™). This is a pluggable feature containing two co-processors. In the case of the z10 BC, feature 0870 provides one co-processor. The co-processors (or 'cards') can provide secure key storage and acceleration when configured in CEX2C mode. The co-processors can provide acceleration only when configured in CEX2A mode. In the configurations shown, the Crypto Express2 feature is used only as an accelerator for the asymmetric cryptographic operations that take place during the SSL handshake, used to establish an SSL session.

Central Processor Assist for Cryptographic Function (CPACF): Enabled by feature code 3863. This is a set of problem-state machine instructions that can accelerate:

1. Symmetric cryptographic operations:

   DES, TDES on z990, z890
   AES-128 on IBM System z9
   AES-192, AES-256 on IBM System z10.

2. MAC hashing algorithms:

   SHA-1 on z990 and z890,
   SHA-256 on IBM System z9
   SHA-384 and SHA-512 on IBM System z10

3. Pseudo Random Number Generator (PRNG) beginning with the IBM System z9.

In the configurations shown, the CPACF is used to accelerate the decryption and encryption of the messages sent during an SSL session, enciphered using DES, triple-DES (TDES or 3DES), or AES.

Hardware Requirements

1. Feature code 3863 enabled on the system (Central Processor Assist for Cryptographic Function enablement)
2. One or more cryptographic processors supplied by Crypto Express2 feature 0863, online to the Linux system. The cryptographic devices can be configured either in accelerator (CEX2A) or coprocessor (CEX2C) mode, but since the solution makes use of acceleration only, the CEX2A mode is preferred. The solution was tested with Linux running under z/VM®, with CRYPTO APVIRT specified in the Linux guest directory entry. This allows sharing of cryptographic devices online to z/VM, among Linux guests.

For more information on making cryptographic hardware available to Linux on System z, see ***Cryptographic Hardware Use Cases for Web Servers on Linux on IBM System z*** at the IBM Platform Test for System z Web site:
http://www.ibm.com/systems/services/platformtest/servers/systemz_library.html#related%20publications

Software Requirements: z/VM

When using a Crypto Express2 configured in coprocessor mode (CEX2C) under z/VM, where the CEX2C is shared (CRYPTO APVIRT specified in the Linux guest directory entry), the fix for z/VM APAR VM64727 is required.

Software Requirements: WebSphere Application Server

The solution was tested with WebSphere V7.0 Network Deployment (64-bit) on Novell SUSE Linux Enterprise Server (SLES-) 10 SP2 with libica and openCryptoki updates.

WebSphere Application Server uses the IBMPKCS11Impl provider supplied with the Java™ JRE/JDK to interface with the PKCS#11 layer to offload cryptographic operations to hardware. This provider maintains its own list of supported hardware:
http://www.ibm.com/developerworks/java/jdk/security/60/secguides/pkcs11implDocs/IBMPKCS11SupportList.html

As stated on that web site, for the Crypto Express2, additional maintenance is needed on the openCryptoki packages that are included with SLES10 SP2. The libica and openCryptoki rpms must be the SLES10 SP3 level or above. The minimum levels are:

For libica:

- libica-1.3.8-0.7.s390x.rpm
- libica-32bit-1.3.8-0.7.s390x.rpm

For openCryptoki:

- openCryptoki-2.2.4-0.12.10.s390.rpm
- openCryptoki-32bit-2.2.4-0.12.10.s390.rpm
- openCryptoki-64bit-2.2.4-0.12.10.s390x.rpm

WebSphere V7.0 and the Java SDK must be at the Fixpack 7 level.

Software Requirements: IBM HTTP Server

The IBM HTTP Server V7 supplied on the WebSphere V7.0 64-bit Supplemental CD image was used in our testing. The minimum Fixpack level required is Fixpack 5, which fixed a problem running the iKeyman utility provided on the 64-bit Supplemental CD.

The IBM HTTP Server uses the IBM Global Security Toolkit (GSKit) to perform cryptographic operations. IHS support for cryptographic hardware is dependent on the support provided by GSKit. See this page for information on GSKit support of cryptographic hardware: http://www.ibm.com/developerworks/tivoli/library/t-gsk7/index.html

As stated on the Web site, GSKit 7, which is used by both IHS V6.1 and IHS V7.0, supports both the Crypto Express2 feature, and the CPACF hardware. IHS V6.1 and V7.0 exploit the GSKit support for using the Crypto Express2 card or accelerating the PreMasterSecret decrypt operation during an SSL handshake on Linux for IBM System z distributions:

- Red Hat Enterprise Linux (4 and 5)
- Novell SUSE Linux Enterprise Server (SLES) 9, 10, and 11.

IHS V7.0 added exploitation of the GSKit support for using the CPACF to accelerate the symmetric encryption and decryption of messages sent during an SSL session. The IHS V7.0 CPACF exploitation testing for this report was done on SLES 10 SP2.

Note: IHS V6.1 has added support for using the CPACF hardware in APAR PK93112.

Cryptographic Hardware Initialization

For both WebSphere Application Server and IBM HTTP Server, the z90crypt cryptographic device driver must be loaded, the PKCS#11 slot daemon (pkcsslotd) must be started, and the IBM Cryptographic Adapter Token, (ICA Token), must be initialized. Perform the following steps:

1. Load the z90crypt device driver and start the PKCS#11 Slot daemon (pkcsslotd) by issuing these two commands:

```
# rcz90crypt start
Loading z90crypt module
                                                              done
```

```
# rcpkcsslotd start
Starting pkcsslotd daemon:usermod: `root' is primary group name.
                                                                done
```

2. Verify the availability of cryptographic hardware

   a) To verify that the cryptographic devices supplied by the Crypto Express2 feature are available, issue the command 'cat /proc/driver/z90crypt'

```
# cat /proc/driver/z90crypt
zcrypt version: 2.1.0
Cryptographic domain: 10
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 0
```

   Note the total device count is 1, and the CEX2A count is 1. This shows that there is at least one processor supplied by the Crypto Express2 feature online to Linux, and the processor is configured in accelerator (CEX2A) mode. There may be more CEX2A devices available, if Linux is running under z/VM. z/VM will distribute requests to multiple cryptographic processors (or adjunct processors – (AP's)) if there are multiple AP's available to z/VM, and 'CRYPTO APVIRT' is specified in the directory entry of the Linux guest. However, Linux under z/VM will 'recognize, at most, one online cryptographic device.

   b) To verify that CPACF is available for the desired cryptographic operations, enter the command: 'icainfo', which is provided by the libica package:

```
# icainfo

The following CP Assist for Cryptographic Function CPACF)
operations are supported by libica on this system:
SHA-1:    yes
SHA-256:  yes
SHA-512:  yes
DES:      yes
TDES-128: yes
TDES-192: yes
AES-128:  yes
AES-192:  yes
AES-256:  yes
PRNG:     yes
```

When the CPACF feature (3863) is active on the machine, the response to the icainfo command displays a 'yes' for all operations on an System z10. Processors prior to this will show a yes for those operations supported by that particular machine type.

3. Initialize the Hardware Cryptographic Token and set the Security Officer and User PINs.

   To initialize the PKCS#11 token and assign a label and set the PINs:

   a) Login with root and run the pkcsconf utility with the -c <slot_number> and -I (upper case letter i) options. When prompted, enter the default Security Officer PIN (87654321). The pkcsconf utility is located in the /usr/sbin directory:

   ```
   # pkcsconf -c 0 -I                          (Initialize the token for slot 0)
   Enter the SO PIN: ******** <-------------------------------------- 87654321
   Enter a unique token label: metlnx21
   ```

   b) Display the token
   Use the pkcsconf –t command to display the token:

   ```
   # pkcsconf -t
   Token #0 Info:
   Label: metlnx21
   Manufacturer: IBM Corp.
   Model: IBM ICA
   Serial Number: 123
   Flags: 0x880445
   Sessions: -1/-1
   R/W Sessions: -1/-1
   PIN Length: 4-8
   Public Memory: 0xFFFFFFFF/0xFFFFFFFF
   Private Memory: 0xFFFFFFFF/0xFFFFFFFF
   Hardware Version: 1.0
   Firmware Version: 1.0
   Time: 18:00:58
   ```

   c) Set the SO (Security Officer) PIN
   Change the SO PIN from the default value (87654321):

   ```
   # pkcsconf -c 0 -P
   Enter the SO PIN: ******** <------------------ 87654321
   Enter the new SO PIN: ******** <-------------- 01234567  (for example)
   Re-enter the new SO PIN: ******** <----------- 01234567
   ```

   d) Set the User PIN

   ```
   # pkcsconf -c 0 -u
   Enter the SO PIN: ******** <------------------- 01234567
   Enter the new user PIN: ******** <------------- 87654321  (for example)
   Re-enter the new user PIN: ******** <---------- 87654321
   ```

   e) Change the User PIN (The User PIN is expired after initial setting)

   ```
   # pkcsconf -c 0 -p
   Enter user PIN: ******** <--------------------- 87654321
   Enter the new user PIN: ******** <------------- 01234567  (for example)
   Re-enter the new user PIN: ******** <---------- 01234567
   ```

   f) Rerun the pkcsconf -t command. It should show the status 'USER_PIN_INITIALIZED' and 'TOKEN_INITIALIZED' :

   ```
   # pkcsconf -t
   Token #0 Info:
   Label: metlnx21
   ```

```
Manufacturer: IBM Corp.
Model: IBM ICA
Serial Number: 123
Flags: 0x44D
(RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|CLOCK_ON_TOKEN|
TOKEN_INITIALIZED)
```

Configuring WebSphere Application Server V7.0 for Linux on IBM System z SLES 10 to use cryptographic hardware for SSL acceleration.

To implement this solution, we change the Java security provider list so that the IBMPKCS11Impl provider is the first provider in the list. The result is that the PreMasterSecret decrypt operation that takes place when a client initiates an SSL session with a WebSphere Server, is offloaded to a cryptographic processor supplied by the IBM Crypto Express2 feature. Also, the message encryption and decryption operations for high strength ciphers (DES, TDES, AES-128, and possibly others depending on the hardware type), are offloaded to the CPACF hardware.

By updating the Java security provider list on a node basis, all of the WebSphere Servers running on the node are enabled to use cryptographic hardware: The Node Agent, all Application Servers, and in the case of a Deployment Manager node, the Deployment Manager. This results in the SSL communication between Application Servers and Node Agents, and Node Agents and the Deployment Manager, being enabled to use cryptographic hardware, as well as the SSL communication between clients and Application Servers.

This solution does not require that the certificates and keys used to establish SSL sessions be located in a Hardware Token, or Hardware Keystore. The hardware acceleration takes place when the SSL session is established using certificates and keys stored in PKCS#12 (p12) keystores.

The steps to implement the solution are:

Download the unlimited jurisdiction policy files and install them in the following location:

/opt/IBM/WebSphere/AppServer/java/jre/lib/security

Complete the following steps to obtain these policy files from the IBM developerWorks® Web site:

a. Go to the following Web site:

http://www.ibm.com/developerworks/java/jdk/security/index.html

b. Click “Java SE 6”.

c. Scroll down and click “IBM SDK Policy files”.

The Unrestricted Java Cryptography Extension (JCE) Policy files for the SDK Web site is displayed.

d. Click “Sign in” and provide your IBM intranet ID and password or register with IBM to download the files.

e. Select the appropriate Unrestricted JCE Policy files and then click “Continue”.

f. View the license agreement and then click “I Agree”.
g. Click “Download Now”.

Perform the actions described by WebSphere APAR PK45677 to force LTPA to use the IBMJCE provider:
http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK45677

For each Server in the cell, add the JVM custom property:
`com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA`
with value true

To set this Custom property for an Application Server, from the WebSphere Administrative Console, go to:
Application Servers > Server > Java and Process Management > Process definition > Java Virtual Machine > Custom properties

Add the Custom property:

com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA	true

This custom property must be set for the Deployment Manager, each node agent, and each Application server in the cell.

1. Create a configuration file to be used by the IBMPKCS11Impl provider.

   The file must be accessible by WebSphere. For example, create the file:

   `/opt/IBM/WebSphere/hwcrypto.cfg`

The contents of the file should look similar to this:

```
#Config file for IBMPKCS11Impl to use Token metlnx21 / slot 0
name = metlnx21
library=/usr/lib/pkcs11/PKCS11_API.so64
##library=/usr/lib/pkcs11/PKCS11_API.so
description=sample config

slotListIndex = 0

disabledMechanisms = {
  CKM_MD5
  CKM_SHA_1
  CKM_MD5_HMAC
  CKM_SHA_1_HMAC
  CKM_SSL3_MASTER_KEY_DERIVE
  CKM_SSL3_KEY_AND_MAC_DERIVE
  CKM_SSL3_PRE_MASTER_KEY_GEN
}
```

Notes:
The name specified (metlnx21) is an arbitrary name that is *set to match the Hardware* Cryptographic Token label. It is not a requirement in this case. But, if this configuration *file were to be* used to access objects in the Hardware Token (using the iKeyman utility that comes with WebSphere 7, for example), the token objects would have the token name pre-pended, which is consistent with how Hardware Token objects are named.

The library specified is the 64-bit PKCS#11 driver supplied by the openCrypt*oki-64bit rpm. We are running* the 64-bit version of WebSphere for Linux on IBM System z, therefor*e the 64-bit PKCS#11 driver is* required.

The slot List Index value matches the slot number of the Hardware Token, which for an ICA *Token on Linux* on IBM System z is slot 0.

The disabled Mechanisms list must match the list shown.

Update the /opt/IBM/WebSphere/AppServer/java/jre/lib/security/java.security provider list to move IBMPKCS11Impl to the top of the list.

Before changing the java.security file, make a backup of the original version.

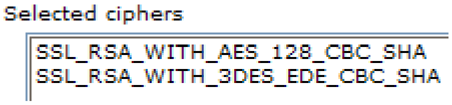
Specify the configuration file created in the previous step on the line defining the IBMPKCS11Impl provider:

```
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
/opt/IBM/WebSphere/hwcrypto.cfg
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
##security.provider.6=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
security.provider.8=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.9=com.ibm.security.sasl.IBMSASL
security.provider.10=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.11=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.12=org.apache.harmony.security.provider.PolicyProvider
```

Note: The line containing the IBMPKCS11Impl provider and hwcrypto.cfg file shown above is one line:

```
security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
/opt/IBM/WebSphere/hwcrypto.cfg
```

Update the Quality of Protection settings in the SSL Configurations used by the WebSphere servers, to use high-strength ciphers. For example, set the list of ciphers to include AES-128 & 3DES, or, just 3DES based on installation requirements:



Selected ciphers

SSL_RSA_WITH_AES_128_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA

The user id that is used to run the WebSphere servers must be added to the pkcs11 group. For example, if the WebSphere servers are running with the use rid wasadm*in, add wasadmin to group* pkcs11:

```
# usermod -G wasgroup,pkcs11 wasadmin
```

Note: Group pkcs11 is created and root is added as a member the first time that the pkcsslotd daemon is started.

Restart the cell.

Verification

Each WebSphere server configured to use cryptographic hardware will open a handle to the z90crypt device driver.

This can be checked by issuing the command:

```
# cat /proc/driver/z90crypt
zcrypt version: 2.1.0
Cryptographic domain: 10
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 1
```

Each SSL handshake done by a WebSphere server configured to use cryptographic hardware will result in the z90crypt 'Per-device successfully completed request' counter being incremented:

```
# cat /proc/driver/z90crypt
zcrypt version: 2.1.0
Cryptographic domain: 10
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 1

Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C
6=CEX2A
0600000000000000 0000000000000000 0000000000000000 0000000000000000

Waiting work element counts
0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

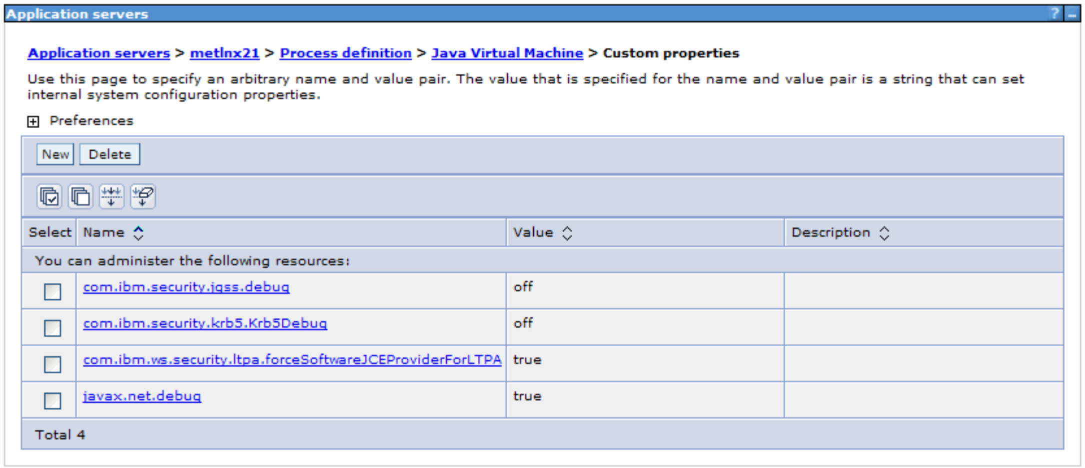
```
Per-device successfully completed request counts
00000000 002310CE 00000000 00000000 00000000 00000000 00000000 00000000
```

There is currently no counter for cryptographic operations offloaded to the CPACF hardware. To verify the use of the CPACF hardware, performance tests can be run with and without the IBMPKCS11Impl provider first in the java.security provider list. The larger the messages requested from WebSphere using SSL, the greater the performance benefit that is obtained using the CPACF. One way to verify with near certainty that the CPACF hardware is being used, is by turning on WebSphere JSSE tracing, and validating that the IBMPKCS11Impl provider is being used to perform the symmetric cryptographic operations.

To turn on JSSE tracing for an Application Server using the Administrative console,

(Make this into a list of steps) go to **Servers** > **Server Types** > **WebSphere application servers** > Click on the server name.
Under Server Infrastructure, expand Java and Process Management.
Click on **Process definition** > **Java Virtual Machine** > **Custom properties**.
Add the Custom property: javax.net.debug with the value true.

For example:



Application servers > metlnx21 > Process definition > Java Virtual Machine > Custom properties

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

Select	Name	Value	Description
You can administer the following resources:			
□	com.ibm.security.jgss.debug	off	
□	com.ibm.security.krb5.Krb5Debug	off	
□	com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA	true	
□	javax.net.debug	true	
Total 4			

After saving, and restarting the application server, an HTTPS request to the application server will result in messages in the SystemOut.log showing the ClientHello and ServerHello:

```
*** ClientHello, TLSv1
........
*** ServerHello, TLSv1
```

Following the ServerHello, there will be a message in the log showing the provider that is used to perform the PreMasterSecret decryption. The message will look similar to:

```
PreMasterSecret: Using cipher for decrypt RSA/SSL/PKCS1Padding from
provider from init IBMPKCS11Impl-metlnx21 version 1.6
```

This message shows that the IBMPKCS11Impl provider is used to perform the PreMasterSecret decryption, which results in the offload of that operation to the crypto card.

Following the PreMasterSecret decrypt messages, there will be READ and WRITE messages to 'Change Cipher Spec'. The 'WRITE Change Cipher Spec' messages will show the cipher and provider that will be used to decrypt and encrypt the messages coming from the client. For example:

```
WebContainer: 0, WRITE: TLSv1 Change Cipher Spec, length = 1
JsseJCE:  Using cipher DESede/CBC/NoPadding from provider TBD via init
CipherBox:  Using cipher DESede/CBC/NoPadding from provider from init
IBMPKCS11Impl-metlnx21 version 1.6
```

These messages show that a DES cipher is being used to decrypt and encrypt the messages coming to and from the client for this SSL session, and that the IBMPKCS11Impl provider is going to be used to perform the cryptographic operations. This is a good indication that these cryptographic operations will be performed by the CPACF hardware.

Configuring IBM HTTP Server V7.0 for Linux on IBM System z (SLES 10) to use cryptographic hardware for SSL acceleration.

IBM HTTP Server V7.0 can also be configured to exploit the cryptographic processors supplied by the IBM Crypto Express2 feature, as well as the CPACF, to accelerate SSL processing. Whereas WebSphere uses the IBMPKCS11Impl Java security provider to interface with the PKCS#11 layer to accelerate cryptographic operations using hardware, the IBM HTTP Server uses the Global Security Toolkit (GSKit) to perform this function.

While IHS 6.1 also supports using the Crypto Express2 feature to accelerate the PreMasterSecret decryption that take place during an SSL handshake, support for using the CPACF to accelerate the symmetric encryption and decryption of the messages themselves is new with IHS 7.0.

IHS requires that the personal certificate and private key used to establish an SSL session with a client, are stored in the hardware cryptographic token, in order to exploit cryptographic hardware. The first step then, in configuring IHS to exploit cryptographic hardware, is to create the personal certificate.

Step 1: Create a Personal Certificate for the IBM HTTP Server, using the Hardware Cryptographic Token

To manage keys and certificates for the IBM HTTP Server, the iKeyman utility is run from the <IHS_SERVER_ROOT>/bin directory.

The utility is a Java X Window application. When the IBM HTTP Server is installed from the WebSphere 64-bit Supplemental CD, the Java used to run iKeyman runs in 64-bit mode. To access hardware cryptographic tokens, the 64-bit PKCS#11 driver supplied by the 64-bit openCryptoki rpm must be specified. This is demonstrated in the following example.

Before iKeyman is run, the gskikm.jar file must be removed from the <IHS_SERVER_ROOT>/java/jre/lib/ext directory. For example:

```
/opt/IBM/HTTPServer/java/jre/lib/ext # rm gskikm.jar
```

Assuming the workstation used to login with ssh to the Linux system is running an X Window server, export the DISPLAY variable on the Linux system:

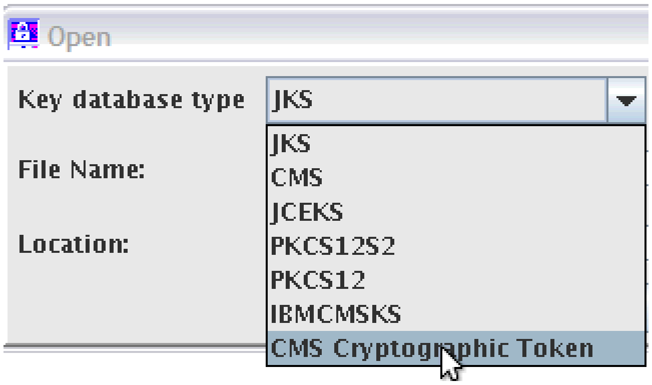
```
/opt/IBM/HTTPServer/bin # export DISPLAY=<IP address of the workstation
used to ssh>:0.0
```

Then, run iKeyman from the <IHS_SERVER_ROOT>/bin directory:

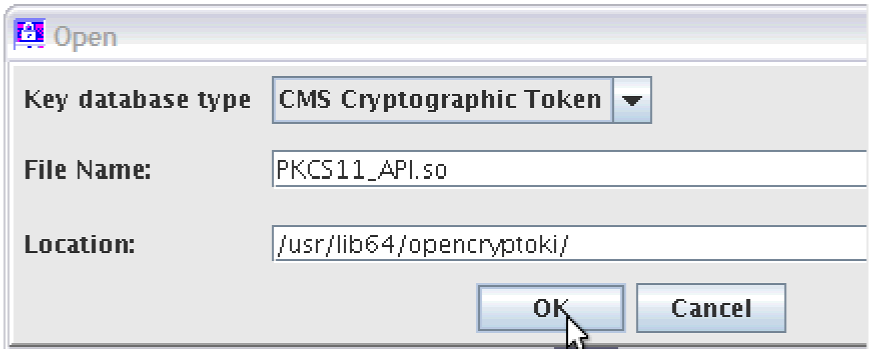
```
/opt/IBM/HTTPServer/bin # ./ikeyman
```

When the iKeyman GUI has initialized, Select **Key Database File: > Open**.

In the Key database type drop-down, select **CMS Cryptographic Token**:



Specify the File Name: PKCS11_API.so and Location: /usr/lib64/opencryptoki
This will load the 64-bit PKCS#11 driver provided by the openCryptoki-64bit package:



Open

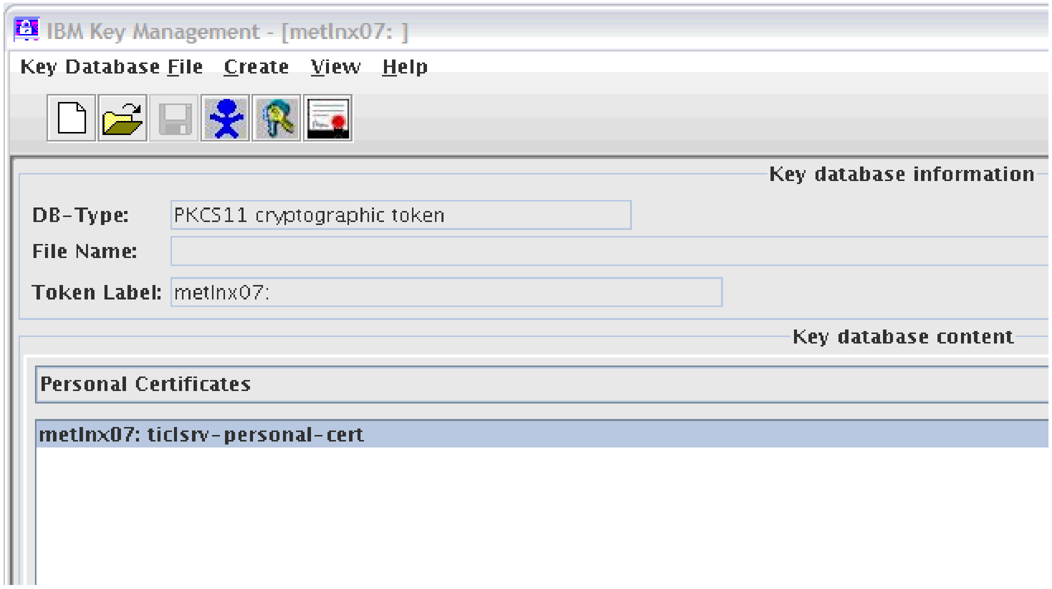
Key database type: CMS Cryptographic Token

File Name: PKCS11_API.so

Location: /usr/lib64/opencryptoki/

On the next panel, the Cryptographic Token Label is already filled in.
Specify the Cryptographic Token Password, using the User PIN value set when the token was initialized.
Clear the 'Open existing secondary key database file' checkbox and click **OK**:

The list of Personal Certificates is displayed:



DB-Type: PKCS11 cryptographic token

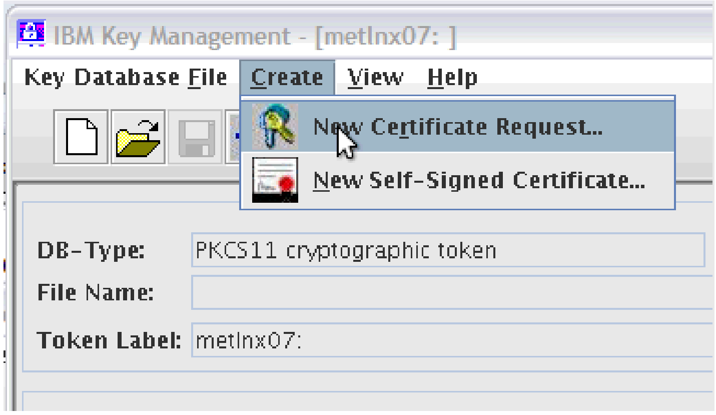
File Name:

Token Label: metlnx07:

metlnx07: ticlsrv-personal-cert

There are several ways to create a Personal Certificate that can be used by the IBM HTTP Server to create SSL sessions with clients. This example creates a certificate request, and then uses the request to have a test certificate generated by a known certificate authority, in this case, Thawte.

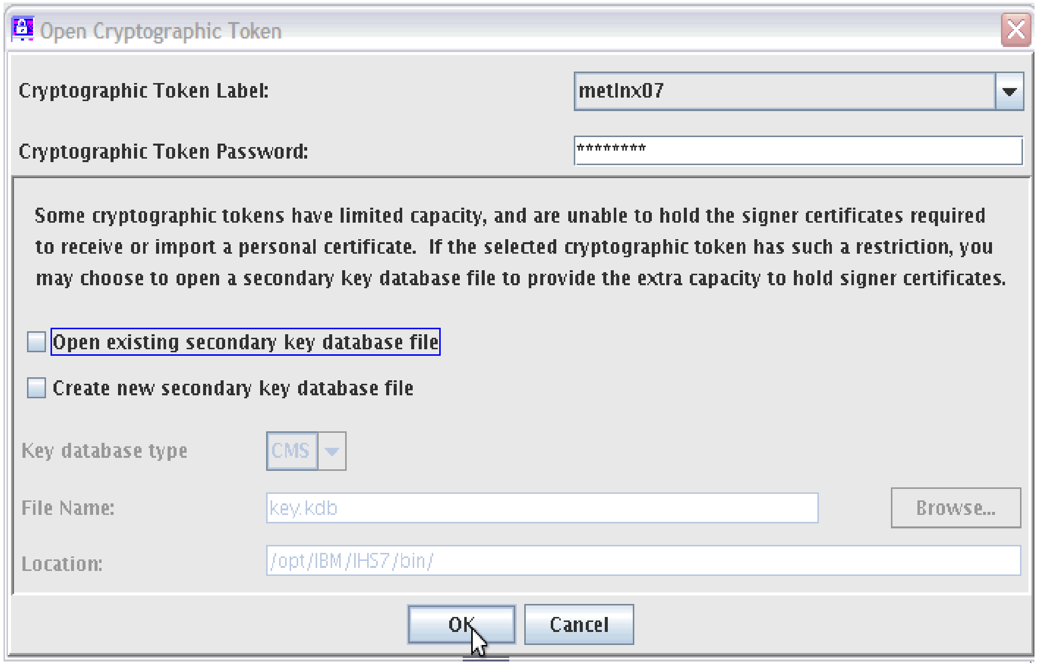
In the iKeyman GUI, with the CMS Cryptographic Token open, at the top of the panel, click **Create**, then click **New Certificate Request**:


IBM Key Management - [metlnx07:]
Key Database File
Create
View
Help
New Certificate Request...
New Self-Signed Certificate...

DB-Type: PKCS11 cryptographic token

File Name:

Token Label: metlnx07:



Open Cryptographic Token

Cryptographic Token Label: metlnx07

Cryptographic Token Password: *********

Some cryptographic tokens have limited capacity, and are unable to hold the signer certificates required to receive or import a personal certificate. If the selected cryptographic token has such a restriction, you may choose to open a secondary key database file to provide the extra capacity to hold signer certificates.

□ Open existing secondary key database file

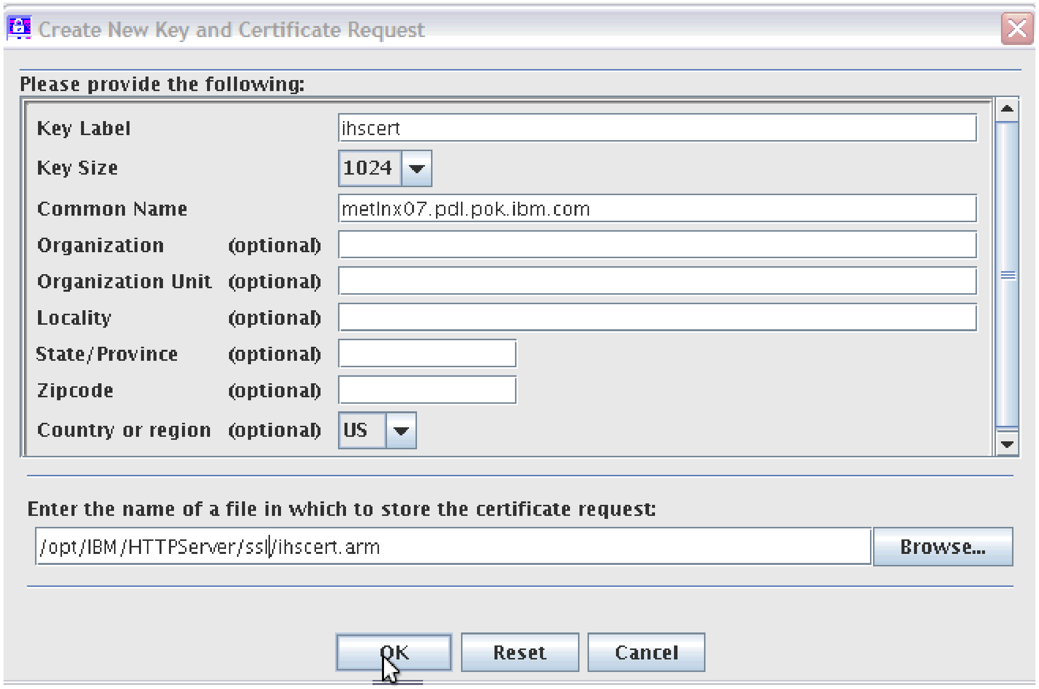
□ Create new secondary key database file

Key database type: CMS

File Name: key.kdb

Location: /opt/IBM/IHS7/bin/

Fill in the details of the request, including the file in which to store the certificate request, and click **OK**:



Create New Key and Certificate Request

Please provide the following:

Key Label		ihscert
Key Size		1024
Common Name		metlnx07.pdl.pok.ibm.com
Organization	(optional)	
Organization Unit	(optional)	
Locality	(optional)	
State/Province	(optional)	
Zipcode	(optional)	
Country or region	(optional)	US

Enter the name of a file in which to store the certificate request:

/opt/IBM/HTTPServer/ssl/ihscert.arm

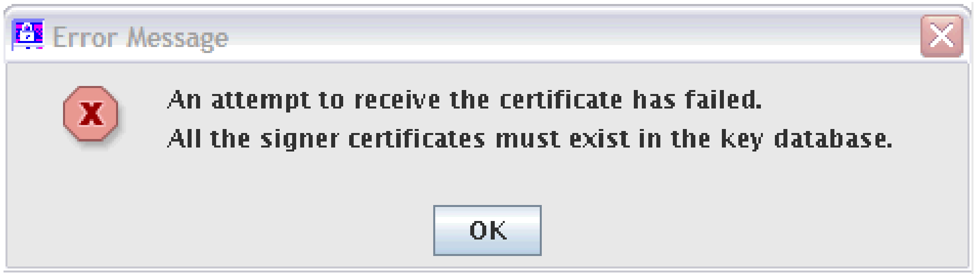
The certificate request, stored in the file /opt/IBM/HTTPServer/ssl/ihscert.arm looks similar to this:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBbzCB2QIBADAwMQswCQYDVQQGEwJVUzEhMB8GA1UEAxMYbWV0bG54MDcucGRs
LnBvay5pYm0uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDxJZVAtORF
HGLSi+iLG9wf2ycgcS68ZESCw186QNouThtMreXwOi5FH+5rrBTSrdzaFQ0T99QK
CoyKKoc2tPnepY85dPxZyKD4OitlJwB3gXwfIPKzM96gK1slMERQW+tZQJC/eT7o
h2Zz4RQK9WuR0og94Z4z/vBuq0teXKsWxwIDAQABoAAwDQYJKoZIhvcNAQEEBQAD
gYEA5QKrNSBnv24VAVktXc7rApbrMquMAv9nPqBoWttKVfuGPsUGWUcTzo8RgnxH
515yh9KBpHjSGhNCH+bd0/xk5RXjQzgXLmfkGOzjO7a4dyGz9tvQcNBHphR0lErl
jRLKDvJCTNYwL0ET2xX7AQyHLWAEUn7xm3NTZSD/8ZNlQEw=
-----END NEW CERTIFICATE REQUEST-----
```

Access the Thawte Web site at http://www.thawte.com, and request a 21-day free trial certificate. After pasting the certificate request into the request window, the personal certificate is generated. Copy and paste it into a file (This example uses a file named ihscert.crt). The contents of this file look similar to the following:

```
-----BEGIN CERTIFICATE-----
MIIC5DCCAk2gAwIBAgIQFwoxlYW9xsOOpTXTDjym5TANBgkqhkiG9w0BAQUFADCB
hzELMAkGA1UEBhMCWkExIjAgBgNVBAgTGUZPUiBURVNUSU5HIFBVUlBPU0VTIE9O
TFkxHTAbBgNVBAoTFFRoYXd0ZSBDZXJ0aWZpY2F0aW9uMRcwFQYDVQQLEw5URVNU
IFRFU1QgVEVTVDEcMBoGA1UEAxMTVGhhd3RlIFRlc3QgQ0EgUm9vdDAeFw0wOTEw
MDgyMTAxMTVaFw0wOTEwMjkyMTAxMTVaMDAxCzAJBgNVBAYTAlVTMSEwHwYDVQQD
ExhtZXRsbngwNy5wZGwucG9rLmlibS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAPEllUC05EUcYtKL6Isb3B/bJyBxLrxkRILDXzpA2i5OG0yt5fA6LkUf
7musFNKt3NoVDRP31AoKjIoqhza0+d6ljzl0/FnIoPg6K2UnAHeBfB8g8rMz3qAr
WyUwRFBb61lAkL95PuiHZnPhFAr1a5HSiD3hnjP+8G6rS15cqxbHAgMBAAGjgaYw
gaMwDAYDVR0TAQH/BAIwADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIw
QAYDVR0fBDkwNzA1oDOgMYYvaHR0cDovL2NybC50aGF3dGUuY29tL1RoYXd0ZVBy
ZW1pdW1TZXJ2ZXJDQS5jcmwwMgYIKwYBBQUHAQEEJjAkMCIGCCsGAQUFBzABhhZo
dHRwOi8vb2NzcC50aGF3dGUuY29tMA0GCSqGSIb3DQEBBQUAA4GBAJJL5gdrhe9w
TnEmnXo1BTYcr2qXyHr+oEJRbHqpWJCTKhcW2aCI4vRbht7i5H1oThfpgl4BRWGQ
v5WLWRkGhdkj10gNEAdkFqsiqEU1UH+k58H/YKKlKjNr4RbuWhGa4qa3xVMton+w
guU3BVZOFmTvy0wD6tm9ZF2oWwrdqIRe
-----END CERTIFICATE-----
```

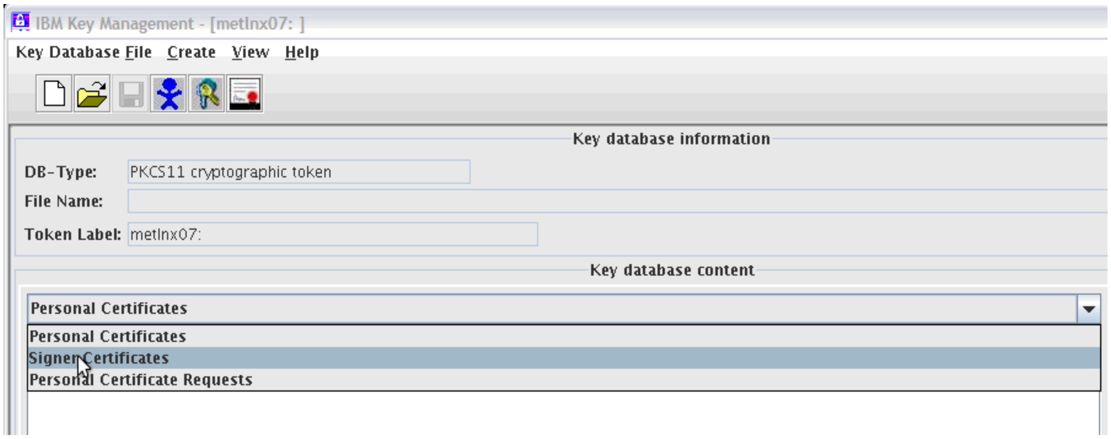
Before receiving the personal certificate into the Hardware Cryptographic Token, first add the Thawte Test Root Certificate to the Token, as a Trusted Signer. If you do not add the Thawte Test Root Certificate before receiving the personal certificate, you will see this error:



An attempt to receive the certificate has failed.
All the signer certificates must exist in the key database.

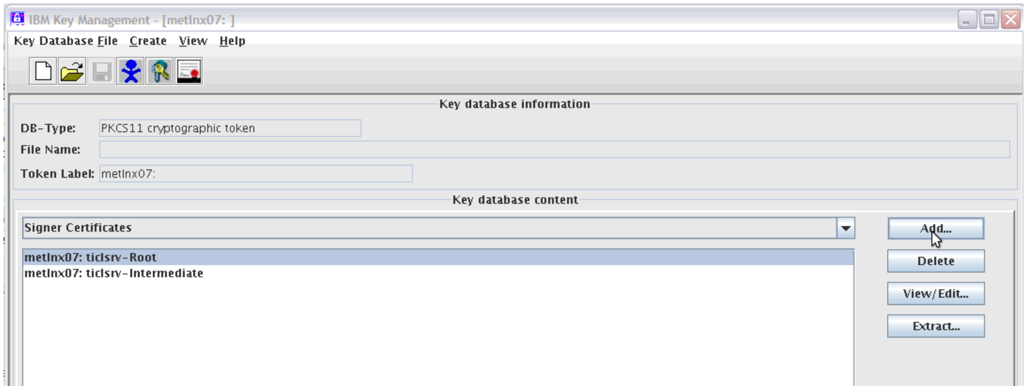
The Signer of a Personal Certificate must be present in the Key Database before the Personal Certificate can be added.

The Thawte Test Root Certificate can be downloaded from the Thawte Web site. To add the The Thawte Root Certificate as a Trusted Signer to the Hardware Cryptographic Token, use the iKeyman GUI to open the **CMS Cryptographic Token**, and select **Signer Certificates** from the Key database content drop-down:



DB-Type: PKCS11 cryptographic token
File Name:
Token Label: metlnx07:

Click the Add button:



DB-Type: PKCS11 cryptographic token

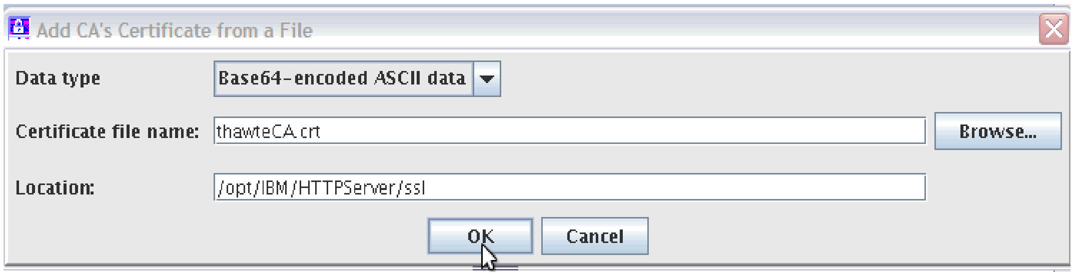
File Name:

Token Label: metlnx07:

Signer Certificates

metlnx07: ticlsrv-Root
metlnx07: ticlsrv-Intermediate

Fill in the Location and File Name fields to point to the file containing the Thawte Test Root Certificate, and click **OK**:

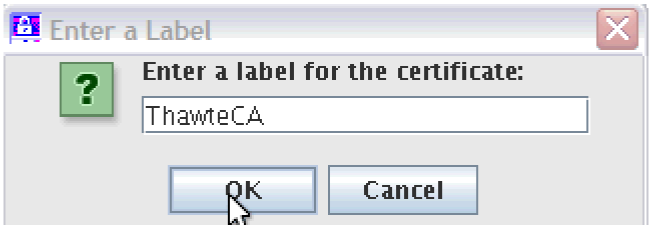


Data type: Base64-encoded ASCII data

Certificate file name: thawteCA.crt

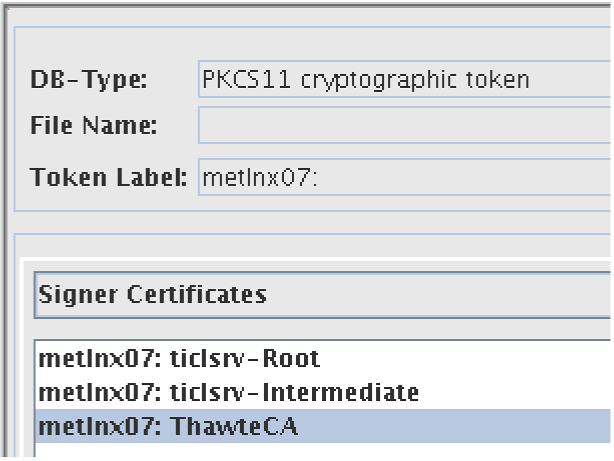
Location: /opt/IBM/HTTPServer/ssl

Enter a label and click **OK**:



Enter a label for the certificate: ThawteCA

The Thawte Test Root Certificate then shows up in the list of Signer Certificates:



DB-Type: PKCS11 cryptographic token

File Name:

Token Label: metlnx07:

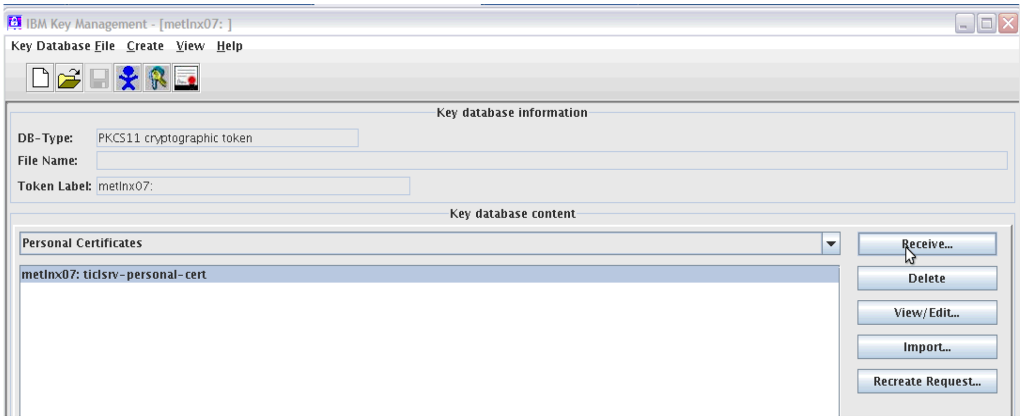
Signer Certificates

metlnx07: ticlsrv-Root

metlnx07: ticlsrv-Intermediate

metlnx07: ThawteCA

Now you can receive the Personal Certificate generated from the Thawte Web site. With the Personal Certificates view of the Key database content showing, click the **Receive** button:



DB-Type: PKCS11 cryptographic token

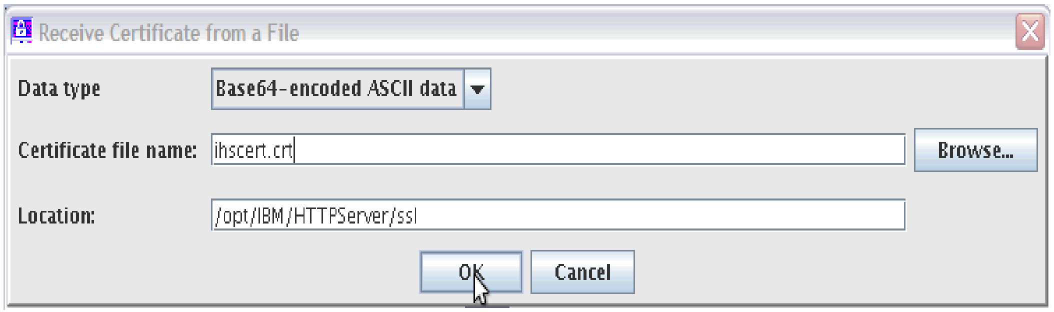
File Name:

Token Label: metlnx07:

Personal Certificates

metlnx07: ticlsrv-personal-cert

Point to the file containing the Personal Certificate generated from the Thawte Web site, and click **OK**:

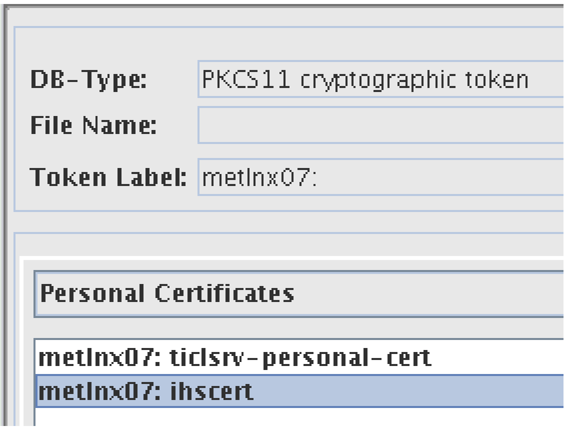


Data type: Base64-encoded ASCII data

Certificate file name: ihscert.crt

Location: /opt/IBM/HTTPServer/ssl

The personal certificate is then displayed in the cryptographic token:



DB-Type: PKCS11 cryptographic token

File Name:

Token Label: metlnx07:

Personal Certificates

metlnx07: ticlsrv-personal-cert

metlnx07: ihscert

Note about importing keys using iKeyman: If iKeyman will be used to import private keys to the CMS Cryptographic Token (from a PKCS#12 file, for example), install the unlimited jurisdiction policy files in the <IHS_SERVER_ROOT>/java/jre/lib/security directory. This will avoid problems when importing keys with a higher level of cryptography than the default policy files permit. See the instructions on obtaining the unlimited jurisdiction policy files in the section: **Configuring WebSphere Application Server V7.0 for Linux on IBM System z SLES 10 to use cryptographic hardware for SSL acceleration.**

Step 2: Update the HTTP Server configuration to enable use of the cryptographic hardware

For the IBM HTTP Server to open the Hardware Cryptographic Token, it must specify the User PIN. IHS accesses the User PIN from a stash file. The stash file is created by running the sslstash utility from the

<IHS_SERVER_ROOT>/bin directory. The following example creates a stash file named pkcs11stash in the /opt/IBM/HTTPServer/ssl directory for the User PIN 01234567:

```
# /opt/IBM/HTTPServer/bin/sslstash -c
/opt/IBM/HTTPServer/ssl/pkcs11stash crypto   01234567
```

Edit the httpd.conf file, and update the HTTPS virtual host to access the cryptographic hardware.

1) Specify the personal certificate 'metlnx07:ihscert' which is the token_label:personal_certificate_label of the personal certificate generated from the Thawte Web site.

2) Specify the SSLStashfile directive, which points to the stash file containing the User PIN.
3) Choose the SSLPKCSDriver directive to specify the 32-bit PKCS#11 driver supplied by the openCryptoki-32bit rpm, because the HTTP Server runs in 32-bit mode:

```
#########################################################
### HTTPS VIRTUAL HOST ##################################
#########################################################
<VirtualHost metlnx07.pdl.pok.ibm.com:443>
ServerName metlnx07.pdl.pok.ibm.com
SSLEnable
SSLProtocolDisable SSLv2
SSLCipherSpec 3A
DocumentRoot /opt/IBM/HTTPServer/htdocs
KeyFile /opt/IBM/HTTPServer/ssl/key.kdb
SSLServerCert  metlnx07:ihscert
SSLStashfile /opt/IBM/HTTPServer/ssl/pkcs11stash
SSLPKCSDriver /usr/lib/pkcs11/PKCS11_API.so
############################
# Symmetric offload
SSLAttributeSet 417 549
############################
</VirtualHost>
##
SSLDisable
SSLV3Timeout 1000
```

Note that in the example above, this directive is also specified:

```
SSLAttributeSet 417 549
```

This directive instructs the HTTP Server to offload symmetric cryptographic operations to the CPACF hardware. The CPACF hardware is utilized for ciphers DES and 3DES on IBM eServer zSeries 990 and 890, plus cipher AES-128 on IBM System z9, plus cipher AES-256 on IBM System z10. Because we have specified the triple-DES cipher be used (SSLCipherSpec 3A), offload of the symmetric cryptographic operations to the CPACF will take place.

Note: IHS 6.1 has added support for symmetric cryptographic processing offload to the CPACF hardware in APAR PK93112. However, the APAR description does not include mention of this feature.

Step 3: Add the User id assigned to the HTTP Server to group pkcs11

The Use rid that is used to run the HTTP Server must be added to group pkcs11. For example, if the user nobody is assigned to run the HTTP Server we would add that user to group pkcs11:

```
# usermod -G nobody,nogroup,pkcs11 nobody
usermod: `nobody' is primary group name.
```

Step 4: Restart the IBM HTTP Server

After restarting the HTTP Server, the command 'cat /proc/driver/z90crypt' will show an additional open handle created to the z90crypt driver:

```
# cat /proc/driver/z90crypt
zcrypt version: 2.1.0
Cryptographic domain: 10
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 1
```

When a client makes an https request to the HTTP Server, the offload of the PreMasterSecret decryption to the cryptographic card results in the request count field in the '/proc/driver/z90crypt' display being incremented:

```
# cat /proc/driver/z90crypt
zcrypt version: 2.1.0
Cryptographic domain: 10
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 2
Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C
6=CEX2A
0000000000000000 0000000000600000 0000000000000000 0000000000000000
Waiting work element counts
0000000000000000 0000000000000000 0000000000000000 0000000000000000
Per-device successfully completed request counts
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
00000000 00000000 00000077 00000000 00000000 00000000 00000000 00000000
```

The offload of the PreMasterSecret decryption occurs at the initialization of an SSL session between a client and the HTTP Server. Once a session is established at the first HTTPS request from a browser, subsequent

HTTPS requests from the same browser reuse the SSL session, and there is no additional offload to a cryptographic card for that particular session. There is, however, offload to the CPACF hardware for each message encryption and decryption operation when using the high strength ciphers listed above. There is currently no display to count the cryptographic operations offloaded to the CPACF.

Notes on the IBM HTTP Server for WebSphere Application Server plug-in

As of this writing, the IBM HTTP Server for WebSphere Application Server plug-in has provided an APAR fix that enables use of the CPACF hardware for acceleration of encrypt and decrypt operations when using SSL to transfer messages between the plug-in and a WebSphere Application Server. The APAR number is PK96110. In addition to installing the new plug-in binaries provided by APAR PK96110, the following Web server plug-in custom properties are required:

SSLPKCSDriver Specifies the fully-qualified name of the loadable module that interfaces with an optional SSL co-processor. The fully-qualified name must include the directory path and the module name.

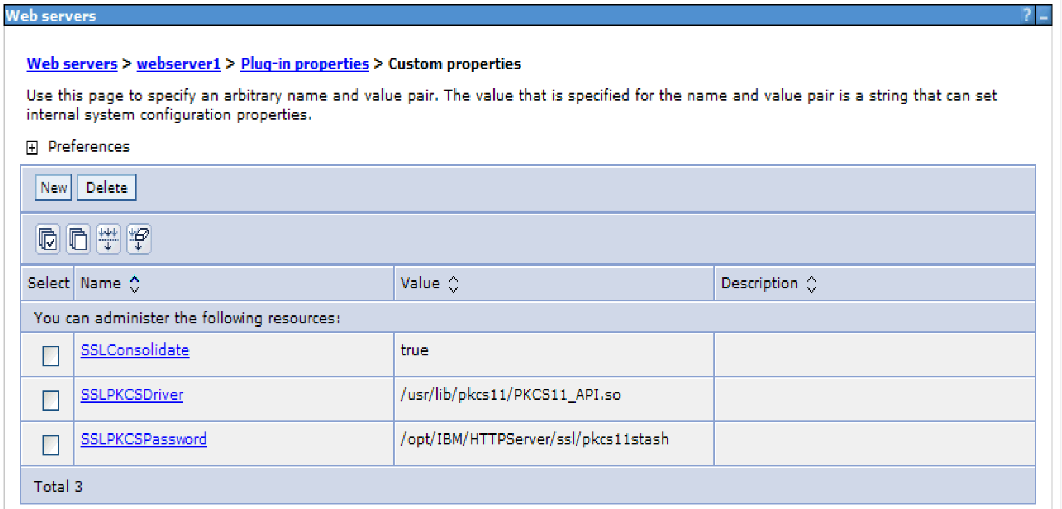
This example uses the name of the 32-bit PKCS#11 driver supplied by openCryptoki: /usr/lib/pkcs11/PKCS11_API.so

SSLPKCS Password Specifies the password for the SSL co-processor with which the module, specified for the SSLPKCS Driver custom property, is interfacing.

This example uses the name of the file containing the stashed User PIN that was created using the sslstash utility from the <IHS_SERVER_ROOT>/bin directory, and which is already specified by the SSLStashfile directive in the httpd.conf: /opt/IBM/HTTPServer/ssl/pkcs11stash.

These custom properties can be set from the WebSphere Administration console. Go to Servers > **Server Types** > **Web servers** > **<webserver>** > **Plug-in properties** > **Custom properties**.

For example:



Web servers > webserver1 > Plug-in properties > Custom properties

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

⊞ Preferences

Select	Name	Value	Description
You can administer the following resources:			
□	SSLConsolidate	true	
□	SSLPKCSDriver	/usr/lib/pkcs11/PKCS11_API.so	
□	SSLPKCSPassword	/opt/IBM/HTTPServer/ssl/pkcs11stash	
Total 3			

When testing with IHS V7 configured to use the WebSphere Application Server plug-in, the problem described by APAR PK82147 was encountered. If you see this error, read this section and follow the steps here.

When customer uses a non-zero value for the ServerIOTimeout option, responses to client requests are intermittently taking an excessive amount of time to be returned to the client. In fact, the responses are being returned when the ServerIOTimeout value pops.

The plugin-cfg.xml file, by default, was generated with non-zero values for ServerIOTimeout. The result was that HTTPS requests being forwarded to the WebSphere Application Server by the Plug-in would hang.

As a work-around, manually update the plugin-cfg.xml file, and changed the ServerIOTimeout values to 0.

For example:

```
<Server CloneID="141hv589f" ConnectTimeout="5" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="metlnx20Node02_metlnx20"
ServerIOTimeout="0" WaitForContinue="false">
```

This value can also be updated using the WebSphere Administrative Console:

a. Click Servers, expand Server Types, and click WebSphere application servers > server_name

b. Under Additional Properties, click Web server plug-in properties.

c. In the Read/Write timeout section, verify that the Use read/write timeout option is enabled, and change the Read/Write timeout value to zero (0) seconds.

At the time of publication, a fix for PK82147 was not yet available.

Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
12/2009
All Rights Reserved

All statements regarding IBM’s future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z

ZSW03139-USEN-00