SSL & TLS Explained: A User Perspective

Glen Brumbaugh

TxMQ

Table of Contents

SSL & TLS Explained: A User Perspective

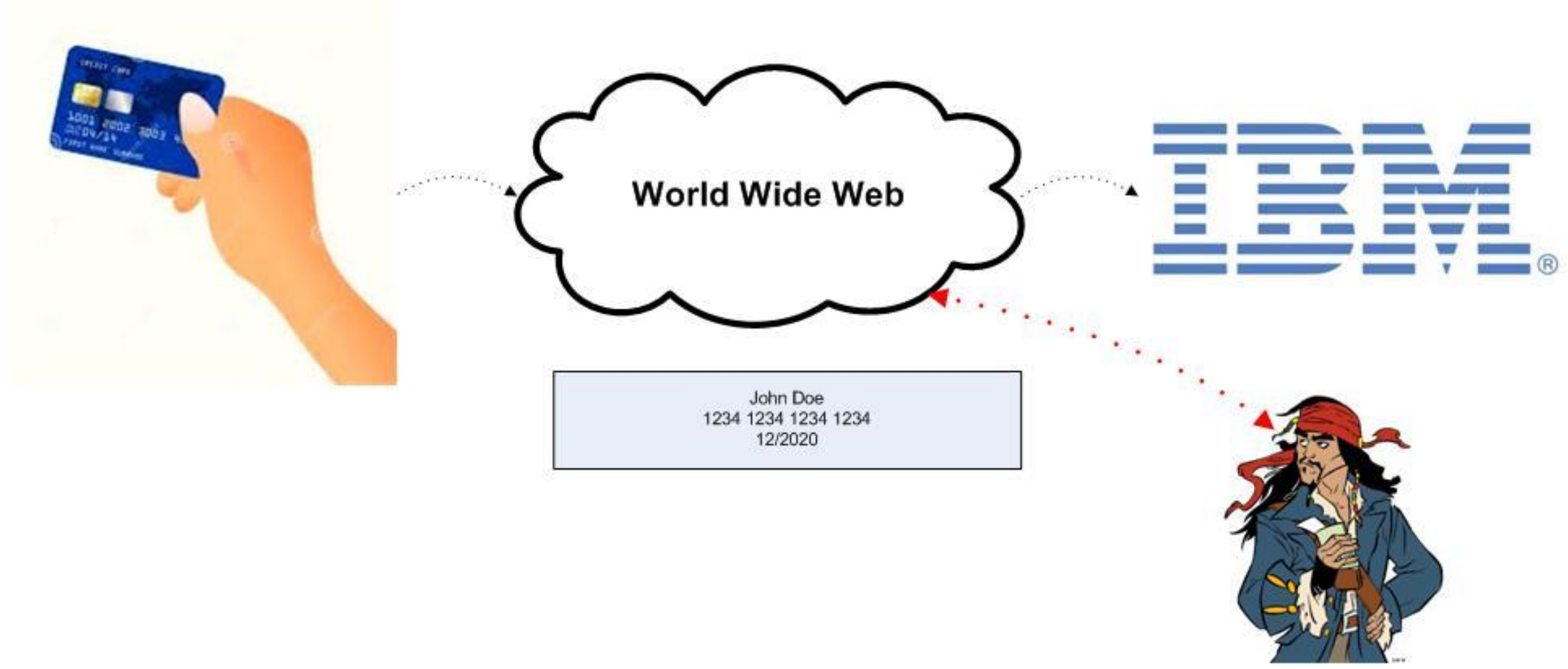
What is SSL & TLS?

SSL & TLS Terminology & History

N
O
T
E
S

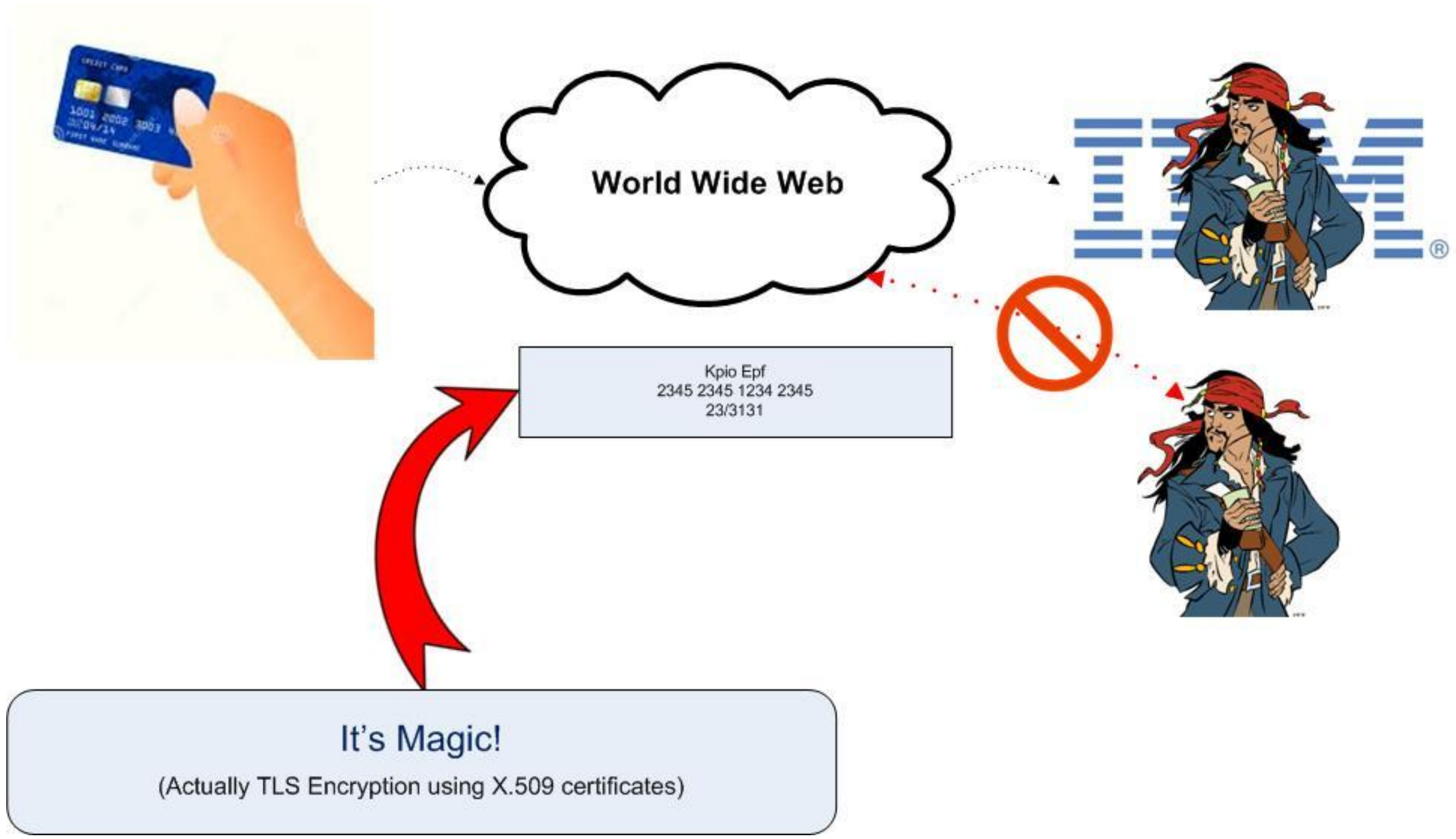
- Terminology
  - Secure Socket Layer (SSL) - A communications protocol
  - Transport Layer Security (TLS) - A communications protocol
  - X.509 - Public Key certificate format standard
  - WMQ Supports both SSL 3.0 and TLS protocols
  - TLS protocols are commonly, and incorrectly, referred to as SSL
- History
  - X.509 introduced in 1988
    - Developed by the International Telecommunications Union (ITU)
  - SSL introduced in 1994
    - Developed by Netscape to support their HTTPS protocol
  - Versions 1.0, 2.0. 3.0 Transport Layer Security (TLS) introduced in 1999
    - Developed by the Internet Engineering Task Force
    - Extended SSL Version 3.0
    - Versions 1.0, 1.1, 1.2, 1.3 (Draft)
  - Each of these versions (both SSL and TLS) is a different protocol
    - These versions do not interoperate!

The Problem perceived by Netscape



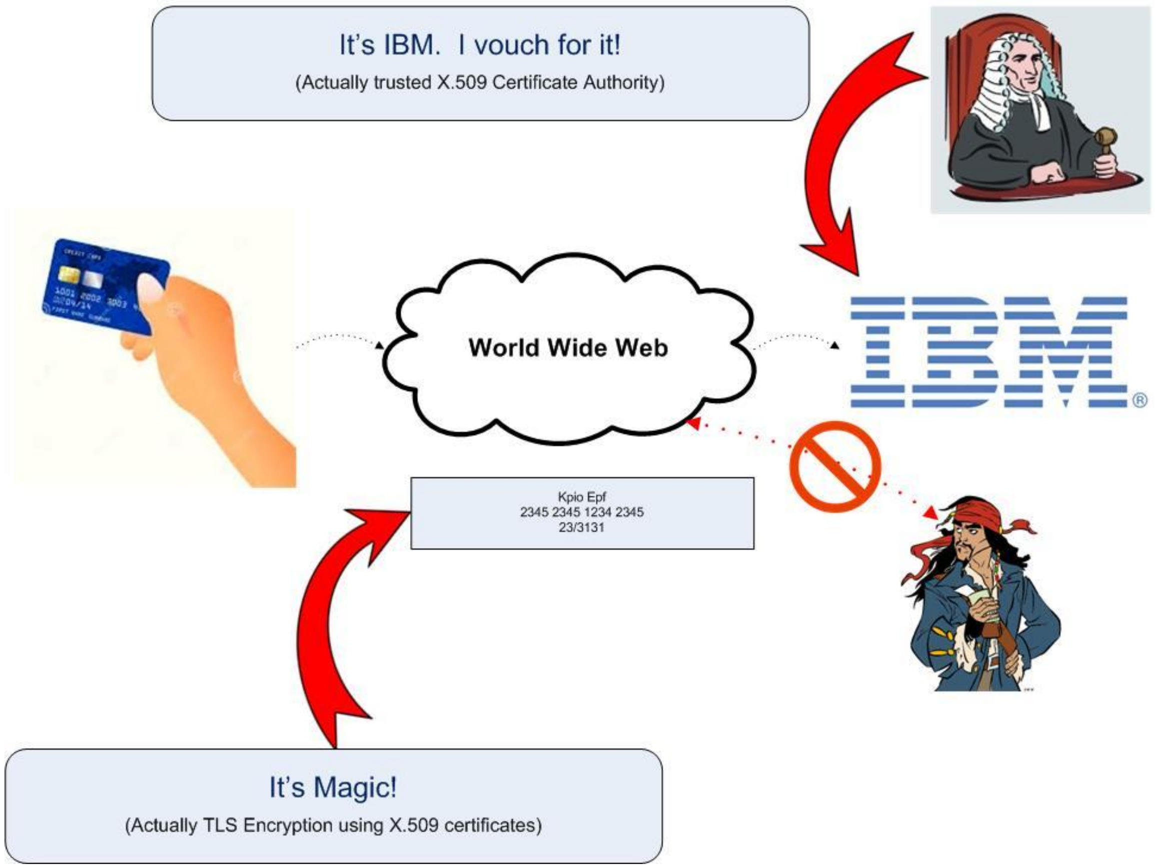
- What’s wrong with this picture?
  - Transmission of sensitive information over an unsecure network (e.g. WWW).
  - Is it really IBM on the other side?

Use of X.509 Certificates to encrypt the data



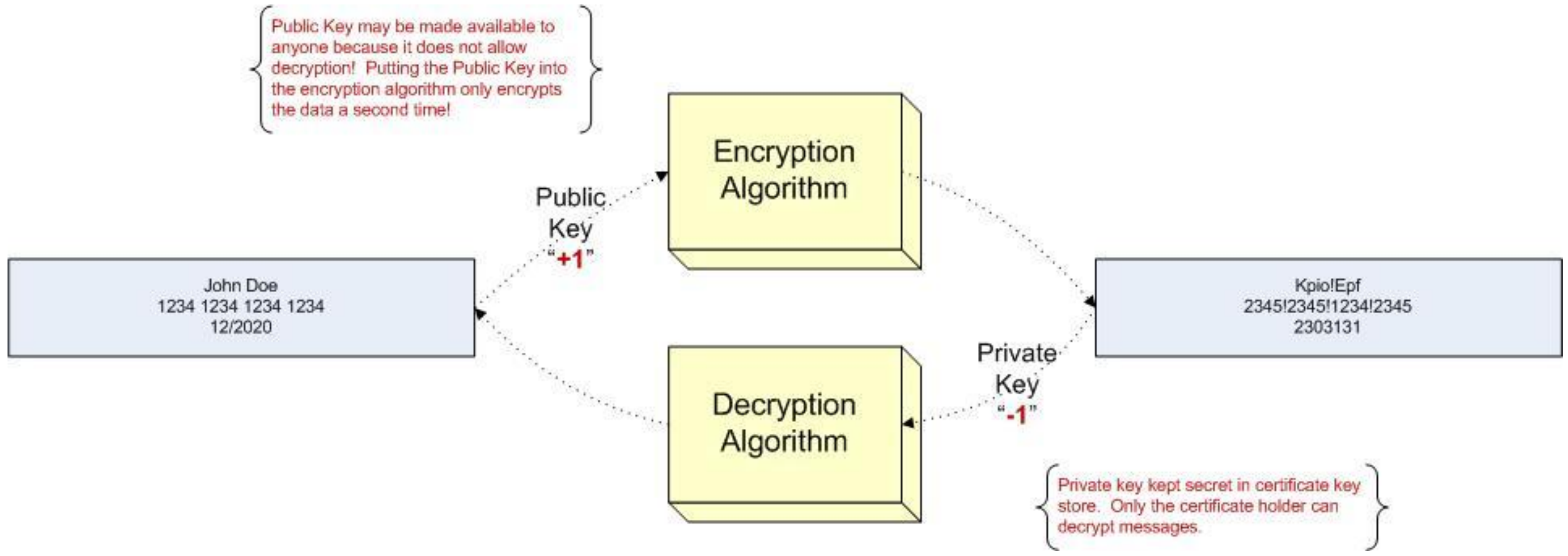
- OK. Better. But you've got me worried. Is it really IBM on the other side?
  - The sensitive data has been encrypted.
  - The destination has not been authenticated.

Use of Certificate Authorities for trust


It's IBM. I vouch for it!
(Actually trusted X.509 Certificate Authority)
World Wide Web
IBM
Kpio Epf
2345 2345 1234 2345
23/3131
It's Magic!
(Actually TLS Encryption using X.509 certificates)

- OK. Fine. I'm ready to send my confidential data.
  - The sensitive data has been encrypted.
  - The destination has been authenticated.

Asymmetrical Encryption


Public Key may be made available to anyone because it does not allow decryption! Putting the Public Key into the encryption algorithm only encrypts the data a second time!
Encryption Algorithm
Public Key "+1"
John Doe
1234 1234 1234 1234
12/2020
Kpio!Epf
2345!2345!1234!2345
2303131
Private Key "-1"
Decryption Algorithm
Private key kept secret in certificate key store. Only the certificate holder can decrypt messages.

- X.509 encryption is based upon a complicated mathematical algorithm.
  - This is a just a trivial example to demonstrate the mechanism.
  - Each character in the incoming message is actually an 8 bit number (0 – 255).
  - In this example, the encoding is in ASCII.
  - The public (encryption) key is added to each character.
  - The private (decryption) key is also added to each character.

How does the Magic work?

N O T E S

- X.509 Certificates are used to store key pairs
  - Public Key, which may be given to anyone, used to encrypt incoming transmissions
  - Private Key, associated with a certificate, used to decrypt incoming transmissions
  - Certificates signed by a trusted Certificate Authority (CA)
- Public Key Exchange is handled by the protocol
  - This is what SSL, and then TLS, were designed to do
    - Each side exchanges it's public key with the other side.
    - Each side may then encrypt traffic being sent to the remote partner
    - The protocol (SSL/TLS) manages the key exchange transparently to the user
- Private Key distribution
  - Web Sites (e.g. IBM.com) using SSL/TLS will obtain a private key from the CA
  - Web Browsers (IE, Safari, Chrome, Firefox)
    - Use generic private server key
      - Provided by the Operating System vendor as part of the OS distribution
      - Used for encryption, does not provide any authentication to merchant
    - Embedded Certificate Authority Public keys
      - All major Certificate Authorities are installed

How does the Magic work? (continued)

N O T E S

- Do you trust the Certificate Authority?
  - Consumers don’t know anything about certificates and Certificate Authorities
  - However, software vendors such as Microsoft and Apple do.
  - The software vendors make this choice for the consumers by preloading CA Public Keys
    - Every major CA Public Key is loaded into the Truststore used by the Browser

- What is the Certificate Authorities function?
  - To ensure that the attributes of the certificate match the source requesting the certificate.
    - For example,
      - The Common Name (or SAN DNS) domain matches the requestor’s domain
      - For example, an IBM.COM certificate should only be issued to an administratively responsible person within IBM.
      - This involves restricting and registering administrators allowed to request certificates for a domain and verifying certificates are requested from the authorized e-mail addresses.

TLS Summary

- SSL was originally developed by Netscape
  - Goal was to enable electronic payment over the WWW.
- TLS encrypts communications between two partners
  - Transmissions are encrypted in both directions over the life of the session.
  - Encryption is managed through X.509 Private certificates being present at both ends
  - Authentication is managed by having the Public Portion of each of the signing CA certificates be present at the opposing end.
- Client Browser validates URL against Private Certificate of website
  - Private certificate must contain the URL or generic domain of the URL
  - Private certificate must be signed by a CA certificate known to the browser
- Server does not validate client Private Certificate
  - No reason to deny access to someone who wants to purchase from you!
- Browsers preloaded with certificates
  - Generic private certificate
  - All major CA certificates (Public Portions)
  - Entire process transparent to Browser user

What does TLS do (and not do)?

What TLS does do (normally)

- Encrypts traffic in both directions
  - This is essential for sensitive data transmitted over the web
  - This may not be relevant to server to server communications within a Data Center
  - TLS connections can be configured to not use encryption
    - NULL_MD5, NULL_SHA (SSL 3.0)
    - TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_NULL_NULL (TLS 1.2)
    - ECDHE_RSA_NULL_SHA256, ECDHE_ECDSA_NULL_SHA256 (TLS 1.2)
- Validates server identity to the client
  - The client validates the server's certificate against the clients Truststore to ensure that the client trusts the identity of the server because the client trusts the signer of the certificate. This is what is called the "web of trust".
  - This is useful for internet financial traffic; you want to know where your money is going
  - This is useful in a data center to protect against "spoofing" a server

What TLS does not do (normally)

- Validate client identity to the server
  - Since this is the primary reason we use SSL/TLS in WMQ, this is a major problem.
    - However, WMQ can be configured to force client authentication on channels
    - SSLCAUTH (REQUIRED)
- Deny access to the server
  - Since this is the primary reason we use SSL/TLS in WMQ, this is a major problem.
    - However, WMQ can be configured to restrict incoming certificates
    - SSLPEER (‘O=IBM, OU=SWG, OU=ISSW’) or SET CHLAUTH
      - SSLPEERMAP SSLPEER (‘O=IBM, OU=SWG, OU=ISSW’) USERSRC(CHANNEL)
    - Note that this requires X.509 certificates to have reasonable Distinguished Name fields
- If you’re not doing both of these things, then maybe you should be!
  - One of the primary uses of SSL/TLS within WMQ is to authenticate clients!
  - If you have encryption turned on, are you also using ALS to protect your messages on disk?

SSL/TLS Summary for WMQ

- SSL or TLS can be used to encrypt traffic between two servers
- SSL or TLS can be used to assert the identity of each of the two servers
  - Identity checking of the client must be requested - SSLCAUTH (REQUIRED)
  - Each server must posses a Personal Certificate in their Keystore
  - Each server must posses the signer certificates of the remote server in their Truststore
  - Note that what is being authenticated is the identity of the server, not the User ID initiating the SSL/TLS connection!
- Neither SSL nor TLS deny access at the protocol level by default
  - However, WMQ requires certificates to be trusted (i.e. their signer in your Truststore)
  - “Filtering” of certificates performed implicitly by the Truststore
  - ”Filtering” of certificates may also be performed explicitly - SSLPEER
- Neither SSL nor TLS deny provide Access Control
  - They are Encryption and Authentication mechanisms
  - WMQ can provide Access Control based upon the authenticaion
    - MCAUSER(*userID*)

Keystore and Certificate Lifecycle

More Terminology - 1

N

O

T

E

S

- Keystore
  - A encrypted and password database to store X.509 certificates
  - Multiple formats for Keystores
    - CMS (Content Management System)
    - JKS (Java Key Store)
    - JCEKS (Java Cryptographic Extension Key Store)
    - PCKS #12 (Public Key Cryptography Standards)
  - Keystores may contain
    - Certificate Signing Requests (CSR)
    - Personal Certificates (Issued by a CA or self-signed)
    - Signer Certificates (Downloaded from a CA or Extracted from a Self-Signed certificate)

- Truststore
  - Two different, but related, meanings:
    - As a file; a specialized Keystore used to only store Signer certificates
    - As a function; the portion of the Keystore file that contains Signer certificates
  - Separate Truststore files are ot used by all software
  - Separate Truststore files simplify certificate management

More Terminology - 2

N
O
T
E
S

- Certificate Signing Request (CSR)
  - A request for a personal certificate generated by a key tool and formatted as a CSR.
  - The CSR is formatted as an encoded text string and may be manipulated as a text string.
  - The CSR is transmitted to a Certificate Authority.
  - The CSR does not contain the Private key. This key remains with the Keystore.
- Certificate Response File
  - The response from a CA to a Certificate Signing Request.
  - The response is an encoded text string similar to a CSR and may be manipulated as a text string.
  - This certificate does not contain either Public or Private keys. These still remain in the Keystore.
- Personal Certificate
  - An X.509 certificate asserting the identify of a URL, Server, or person.
  - Contains both Public and Private keys.
  - Either Self-signed or issued by a Certificate Authority.
  - Multiple formats for certificates:
    - ARM, DER, PEM, PKCS #7, & PKCS #12.
  - Certificate is password protected.

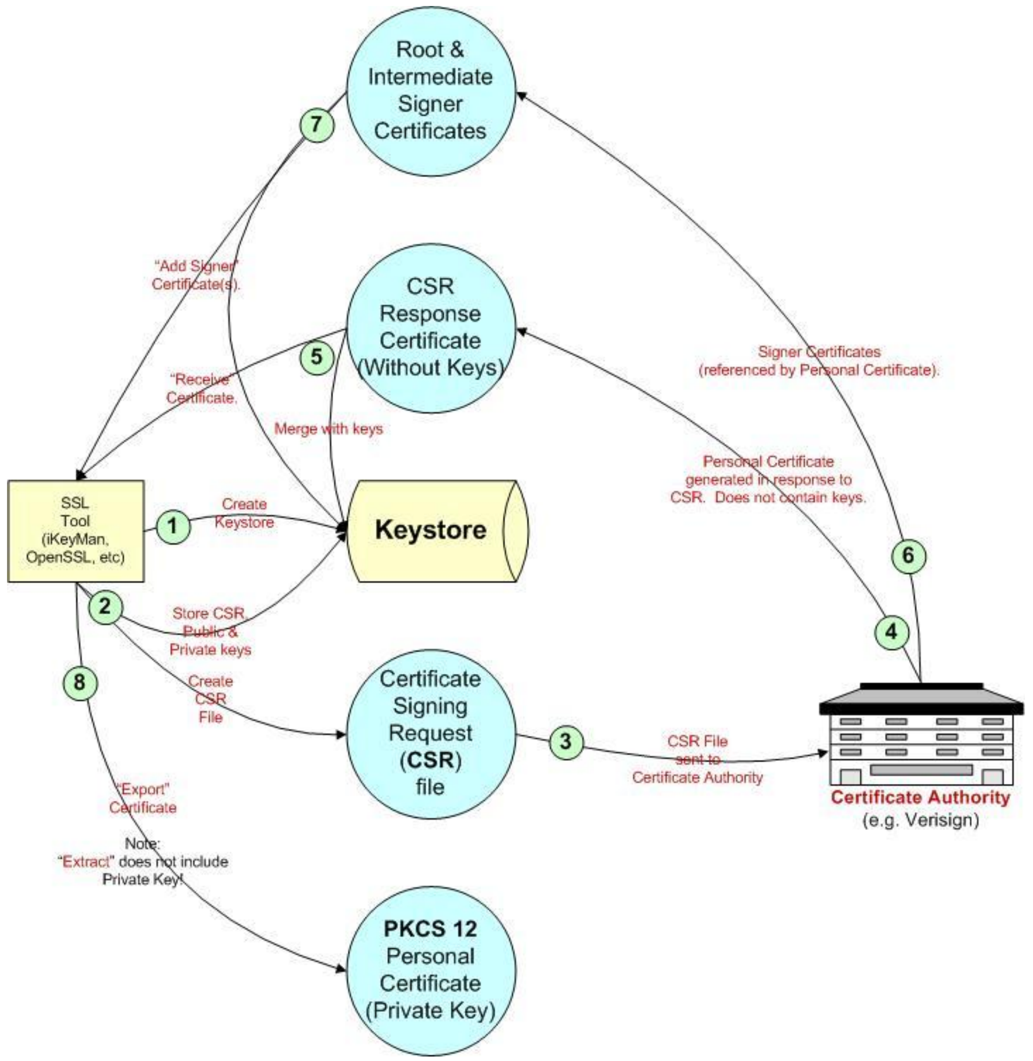
More Terminology - 3

N O T E S

- Signer Certificate
  - An X.509 certificate used for authenticating another certificate
  - Either Self-signed or issued by a Certificate Authority.
  - Contains only the Public Key for the Signer Certificate.

- Public Certificate
  - A subset of a Personal Certificate containing only the Public Key.
  - This public portion of a Personal certificate is what is exchanged during the SSL handshake.
  - This certificate does not require a password.

Keystore Lifecycle


Root & Intermediate Signer Certificates
CSR Response Certificate (Without Keys)
SSL Tool (iKeyMan, OpenSSL, etc)
Keystore
Certificate Signing Request (CSR) file
PKCS 12 Personal Certificate (Private Key)
Certificate Authority (e.g. Verisign)
1 Create Keystore
2 Store CSR, Public & Private keys
Create CSR File
3 CSR File sent to Certificate Authority
4 Personal Certificate generated in response to CSR. Does not contain keys.
5 "Receive" Certificate.
Merge with keys
6 Signer Certificates (referenced by Personal Certificate).
7 "Add Signer" Certificate(s).
8 "Export" Certificate
Note: "Extract" does not include Private Key!

- 1) Create a Keystore
- 2) Request a Personal Certificate
- 3) Transmit CSR to CA
- 4) CA transmits CSR Response
- 5) Receive response into Keystore
- 6) Download signer certificates from CA
- 7) Add or Populate Signer certificates into Keystore
- 8) Export Personal certificate (if necessary to deploy certificate to remote Keystore)

A Note on Passwords

- All Keystores and Truststores require a password
- All Personal Certificates containing private keys require a password
- This can add up to a lot of passwords
  - Some form of password management is needed
- These passwords must be supplied when opening a Keystore or Certificate
  - Can be entered interactive by human users
  - Can be “stashed” in a file for use by software (e.g. WMQ)
    - This is called a “Stash” file

Keystore Summary for WMQ

- Some software has built-in software to manage Keystores / Truststores
  - WebSphere
- Some software requires external software to manage Keystores / Truststores
  - WebSphere MQ
  - WebSphere Message Broker / IBM Integration Bus
  - Multiple certificate mangement software options
    - iKeyMan (IBM Key Manager)
    - OpenSSL
- Multiple Keystore/Truststore formats
  - CMS, JKS, JCEKS, PCKS #12
  - WMQ requires CMS format Keystores
  - WebSphere and Message Broker require JKS Keystores
  - WebSphere Message Broker optionally uses a separate Truststore
- Keystores / Truststores may be built in place or transmitted
  - Built and stored on server where they will be used (requires a Key Tool such as iKeyMan)
  - Built centrally and shipped to destination server

SSL & TLS Explained: A User Perspective

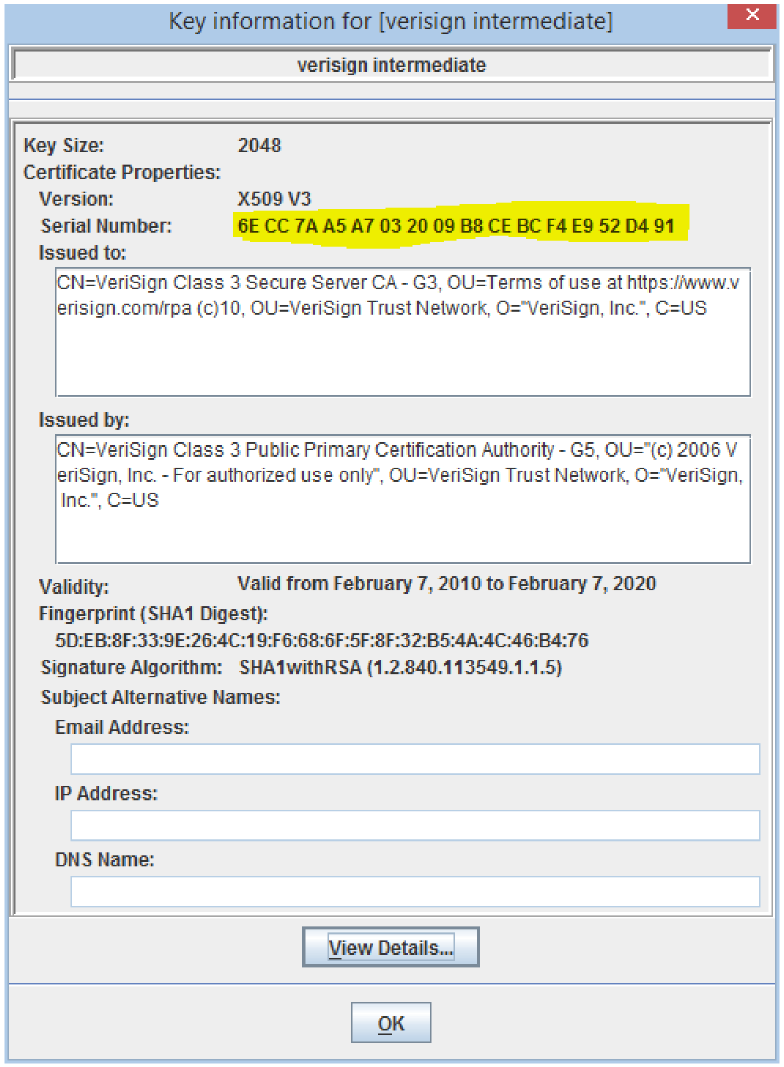
Certificates

X.509 Common Certificate Fields

N
O
T
E
S

- CN (Common Name)
  - Name certificate represents (e.g. "www.capitalware.com")
  - What you see in your browser when you click on a certificate
- O (Organization)
  - Name certificate represents (e.g. "Capitalware")
- OU (Organizational Unit)
  - Hierarchy within organization (e.g. "OU=Education", "OU=MQTC")
- L (Locality)
  - Geographic Location or City (e.g. "Sandusky")
- ST (State)
  - State of Locality (e.g. "Ohio")
- C (Country)
  - Country of Locality (e.g. "US")
- SAN (Subject Alternative Names)
  - E-mail, IP, or DNS names that are aliases of the Common Name (CN)

X5.09 VeriSign certificate in iKeyMan



Key information for [verisign intermediate]

verisign intermediate

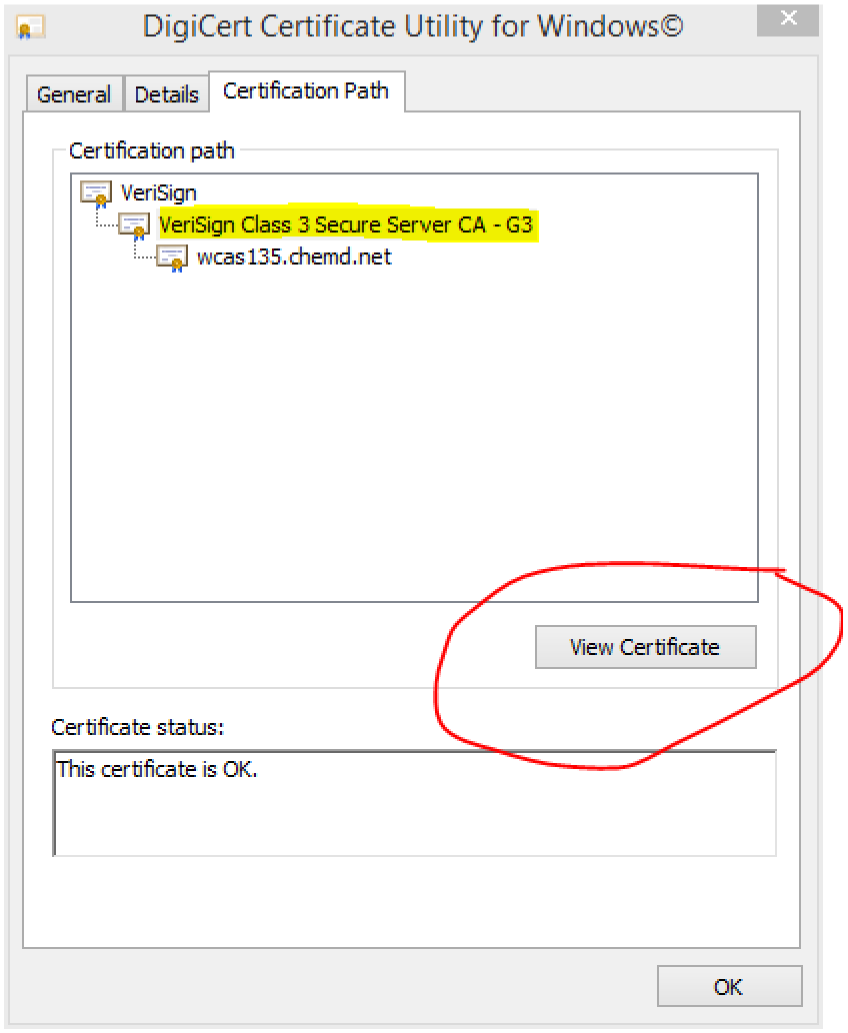
```
Key Size:                  2048
Certificate Properties:
  Version:                 X509 V3
  Serial Number:           6E CC 7A A5 A7 03 20 09 B8 CE BC F4 E9 52 D4 91
  Issued to:
    CN=VeriSign Class 3 Secure Server CA - G3, OU=Terms of use at https://www.v
    erisign.com/rpa (c)10, OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US

  Issued by:
    CN=VeriSign Class 3 Public Primary Certification Authority - G5, OU="(c) 2006 V
    eriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign,
    Inc.", C=US

  Validity:                Valid from February 7, 2010 to February 7, 2020
  Fingerprint (SHA1 Digest):
    5D:EB:8F:33:9E:26:4C:19:F6:68:6F:5F:8F:32:B5:4A:4C:46:B4:76
  Signature Algorithm:   SHA1withRSA (1.2.840.113549.1.1.5)
  Subject Alternative Names:
    Email Address:
    IP Address:
    DNS Name:
```

- iKeyMan tool (IBM Key Manager)
- Display of a VeriSign Intermediate Signing certificate.
- Note the Serial Number. It is this field, and not the Common Name (CN) that identifies this certificate in the signing chain!

Certificate Signing Chain in DigiCert


DigiCert Certificate Utility for Windows©

General | Details | Certification Path

Certification path

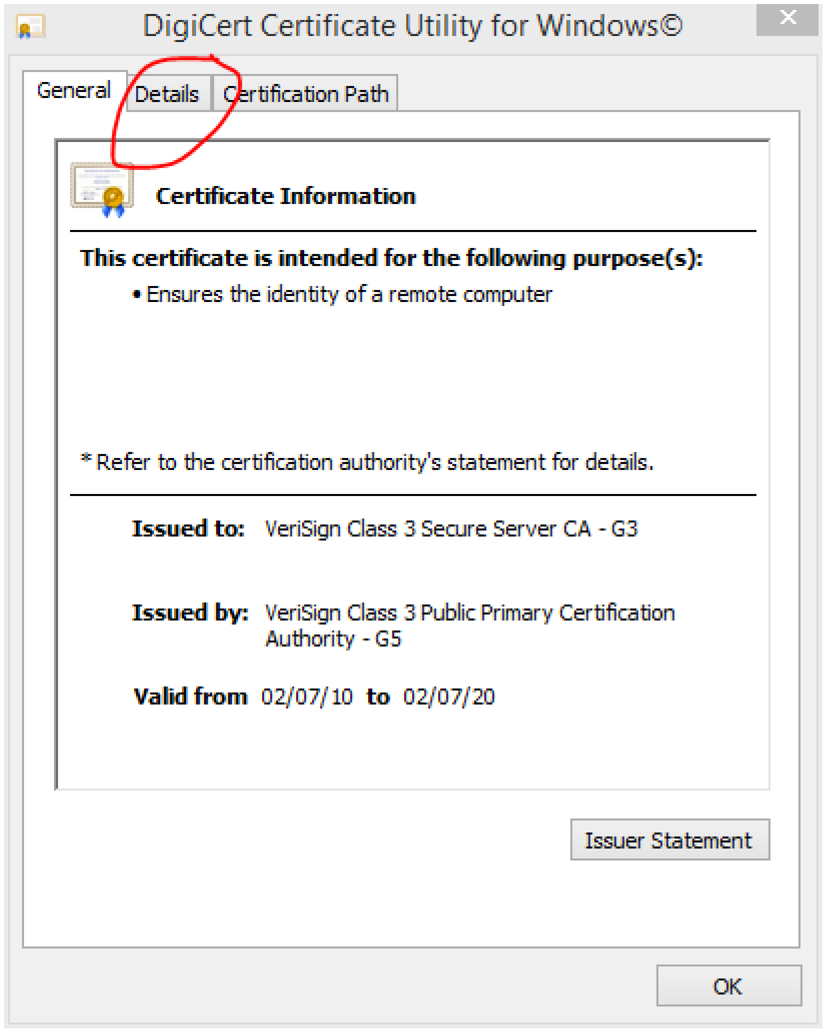
- VeriSign
  - VeriSign Class 3 Secure Server CA - G3
    - wcas135.chemd.net

Certificate status:

This certificate is OK.

- DigiCert tool
- Displays the signing chain for a certificate. In this case, the certificate is for "wcas135.chemd.net".
- Note this (wcas135.chemd.net) is a public server that can be found using "nslookup".
- The certificate with the Common Name of "wcas135.chemd.net" was signed by an Intermediate VeriSign certificate with the Common name of "VeriSign Class 3 Secure Server CA – G3".
- The Intermediate VeriSign certificate was signed by a root VeriSign certificate.

Certificate Signing Chain in DigiCert



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer

* Refer to the certification authority's statement for details.

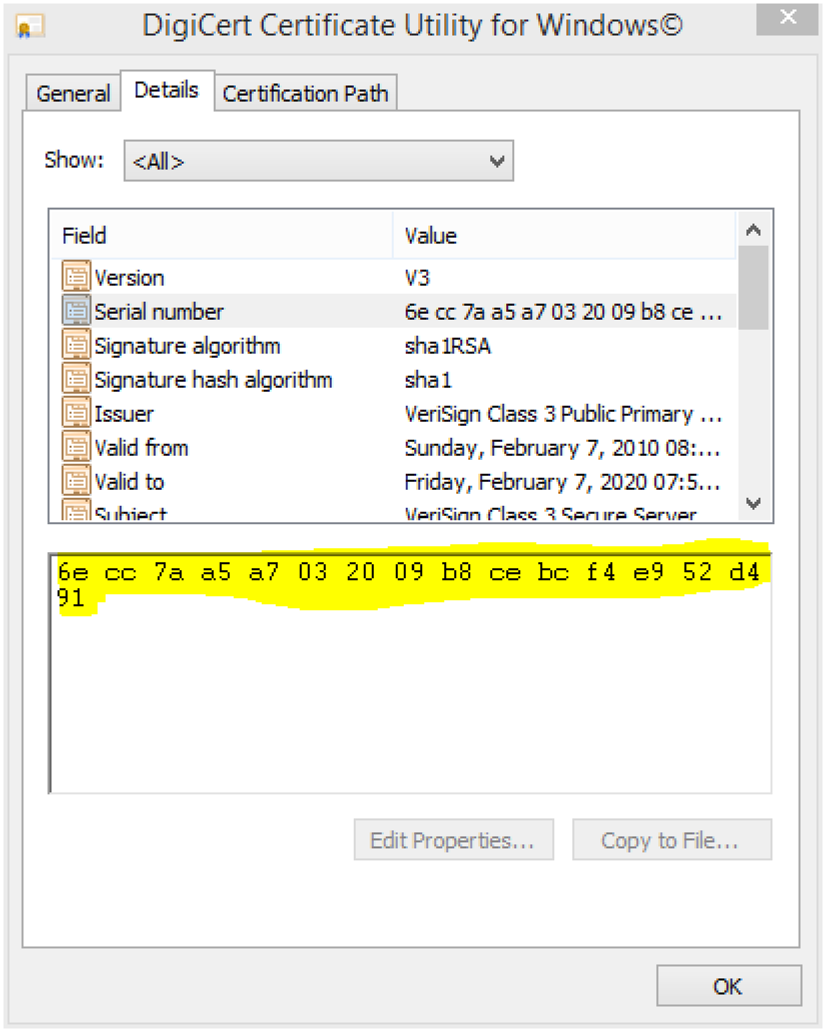
Issued to: VeriSign Class 3 Secure Server CA - G3

Issued by: VeriSign Class 3 Public Primary Certification Authority - G5

Valid from 02/07/10 **to** 02/07/20

- DigiCert tool
- Displays information about the "VeriSign Class 3 Secure Server CA – G3" certificate.
- Note this "Intermediate Signer"certificate was signed by the "VeriSign Class 3 Public Primary Certification Authority – G5" "Root Signer" certificate.
- Now lets look at the certificate details.

Certificate Signing Chain in DigiCert


DigiCert Certificate Utility for Windows©

General | Details | Certification Path

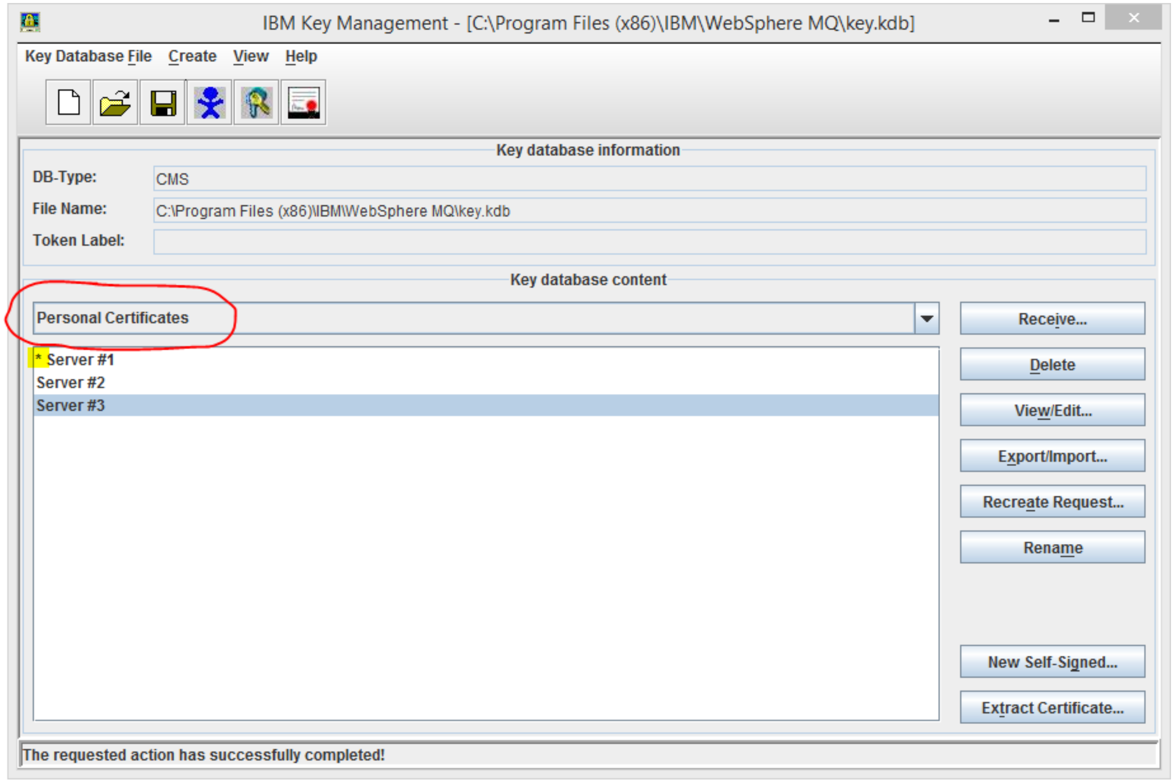
Show: <All>

Field	Value
Version	V3
Serial number	6e cc 7a a5 a7 03 20 09 b8 ce ...
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	VeriSign Class 3 Public Primary ...
Valid from	Sunday, February 7, 2010 08:...
Valid to	Friday, February 7, 2020 07:5...
Subject	VeriSign Class 3 Secure Server

```
6e cc 7a a5 a7 03 20 09 b8 ce bc f4 e9 52 d4
91
```

- DigiCert tool
- Note the Serial Number of the certificate.
- If a remote server needed the Signer certificate for “wcas135.chemd.net”, they would need to ensure that they certificate they imported into their Keystore had this Serial Number.
- Important Note: You must ensure the Serial numbers are correct for your Signer certificates! Some of the Certificate Authorities have multiple certificates with the same Common Name!

Certificate Labels


IBM Key Management - [C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb]

Key Database File Create View Help

Key database information

DB-Type: CMS

File Name: C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb

Token Label:

Key database content

Personal Certificates

* Server #1
Server #2
Server #3

The requested action has successfully completed!

- iKeyMan tool (Personal Certificates)
- The names displayed are Labels, not Common Names (CN)
- The asterisk ("*") denotes the "Default" certificate for the Keystore

Using Certificate Labels

- If there is more than one Personal Certificate, which one should be used?
  - There are three mechanisms.
    - Restrict the Keystore to containing exactly one Personal Certificate
      - No possible confusion
      - Not as flexible
      - ✓ This is the mechanism used by WMB/IIB
    - Use the “Default” certificate
      - Requires a Personal Certificate to be identified as the “Default” certificate
      - Easy to make mistakes
    - Find the certificate by its label
      - No possible confusion
      - Requires certificates to be identified by a specific label
      - ✓ This is the mechanism used by WMQ
        - ibmwebspheremq*qmgrname*   (All in lower case)
  - These mechanisms are part of the APIs used to access the Keystores
  - The software accessing the Keystore determines the method to use

Certificate Summary for WMQ

- WMQ needs to know where the Keystore is located
  - Queue Manager parameter SSLKEYR
    - Specify the path and filename, but the file extension (“.kdb”)
- WMQ requires that Keystore be in the CMS format
  - *Stored as multiple files*
    - *Keystore file required (must have the ”.kdb” file extension)*
    - *Keystore file required for CSRs (must have the “.rdb” file extension)*
    - *Password Stash file required (must have the “.sth” file extension)*
  - *All files must have the same name, with a differing file extenstion!*
- WMQ requires the certificate to have a specific label in the Keystore
  - ibmwebspheremq*qmgrname*
- WMQ does not care about any of the fields in a certificate
  - Unless the SSLPEER parameter is set!

SSL & TLS Explained: A User Perspective

Certificate Management

Certificate Management Steps

1) Build Keystores and Truststores (One for every server)
   - Build at one location using central tool and distribute Keystores and Truststores to each server
   - Build on each server using a tool on each server
   - Create Stash files for each Keystore

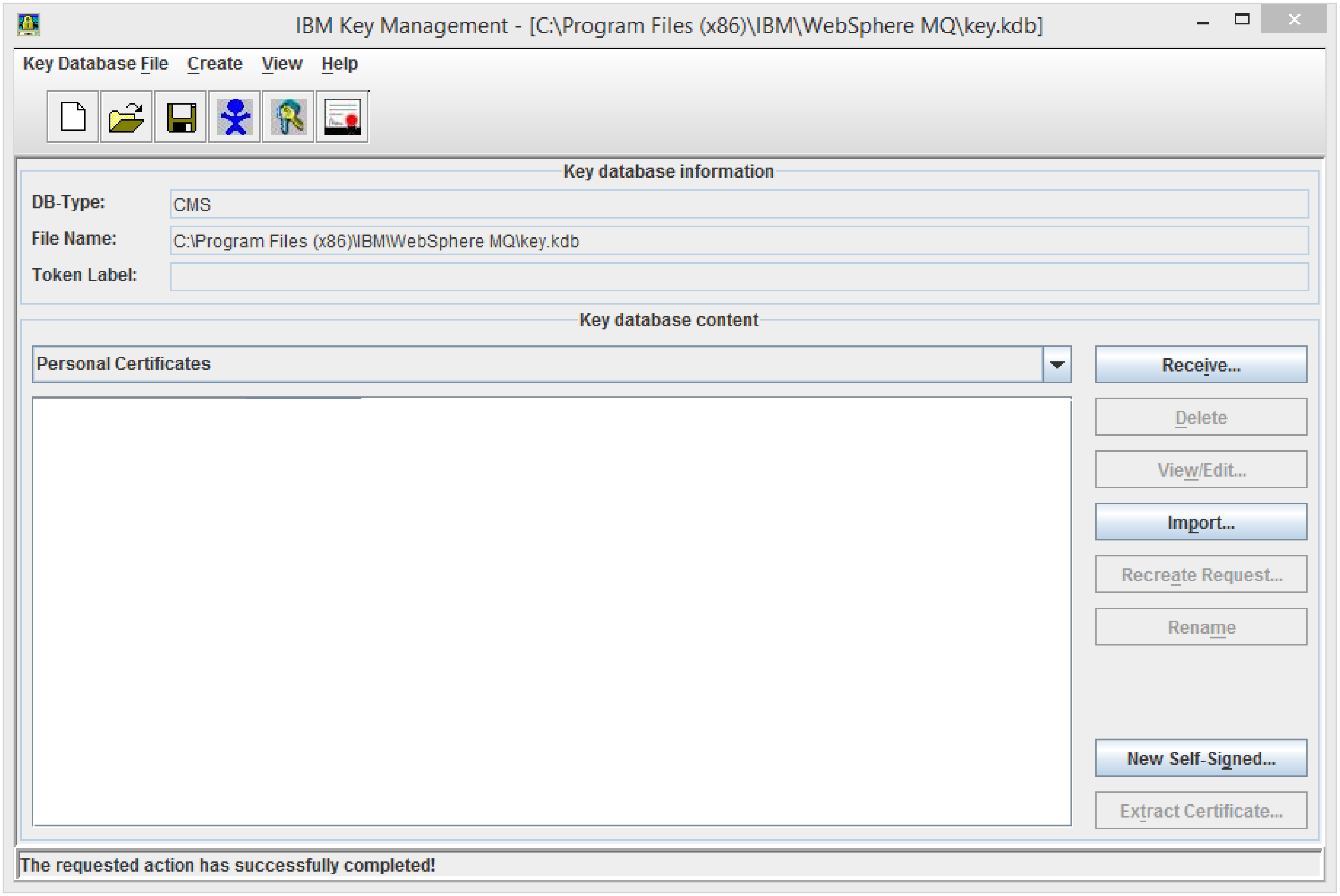
2) Obtain Personal Certificates (One for every server)
   - Generate CSRs to CA or create Self-Signed certificates
   - "Receive" CA Response certificates into Keystore (marries Public Certificate to Private Key)

3) Added Signer certificates to Keystores
   - Adding as few Signers as possible increases security
   - Adding as few Signers as possible increases work (what Signers do I need)
   - Add Signers for all expected incoming certificates (e.g. Certificates from remote servers)
   - "Extract" Public portion (Signer) of Self-signed certificates

4) Distribute Keystores and Truststores to servers

5) Configure software to use SSL/TLS

iKeyMan Home Screen


IBM Key Management - [C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb]

Key database information

DB-Type: CMS

File Name: C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb

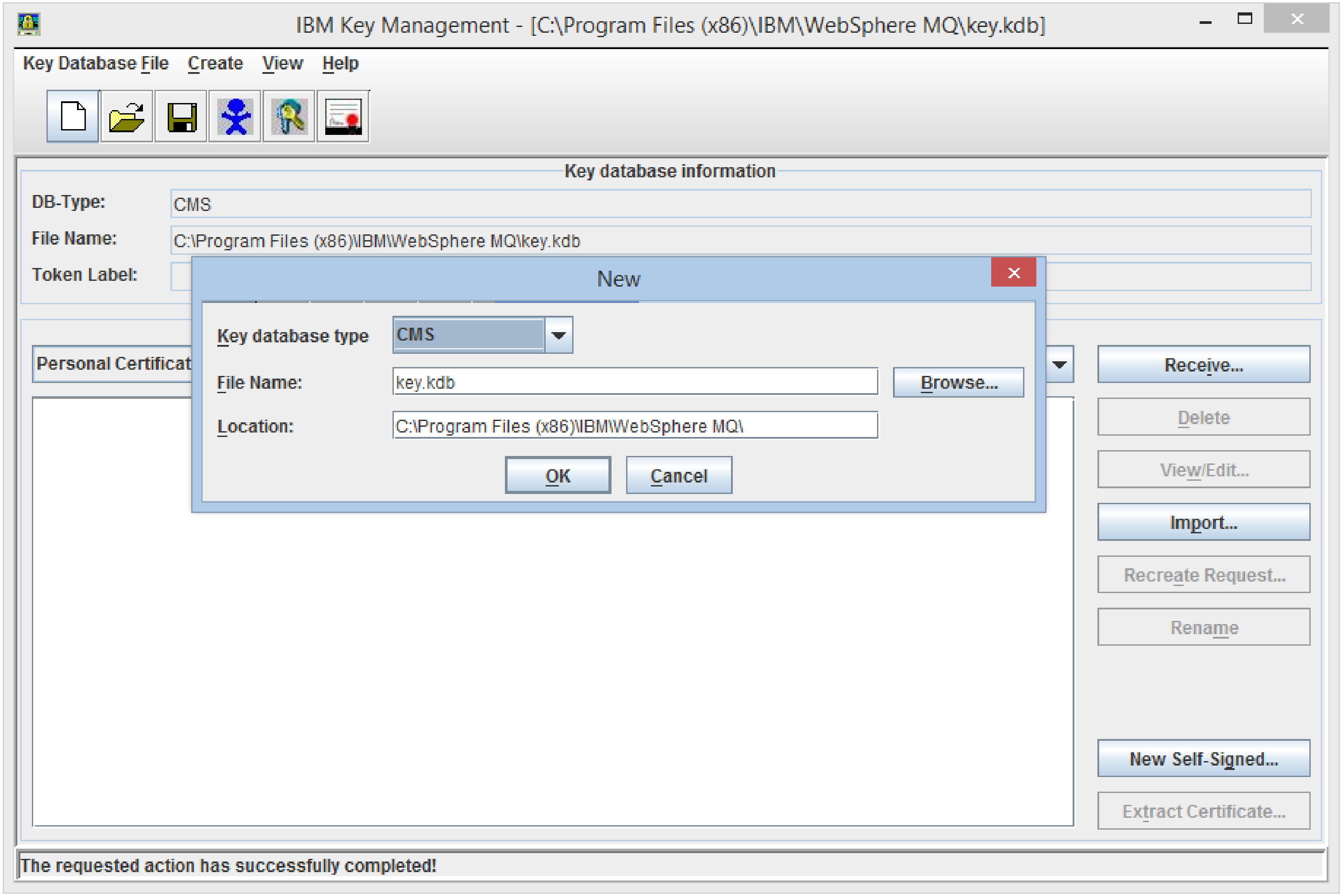
Token Label:

Key database content

Personal Certificates

The requested action has successfully completed!

iKeyMan New Keystore -1



IBM Key Management - [C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb]

Key database information

DB-Type: CMS

File Name: C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb

Token Label:

New

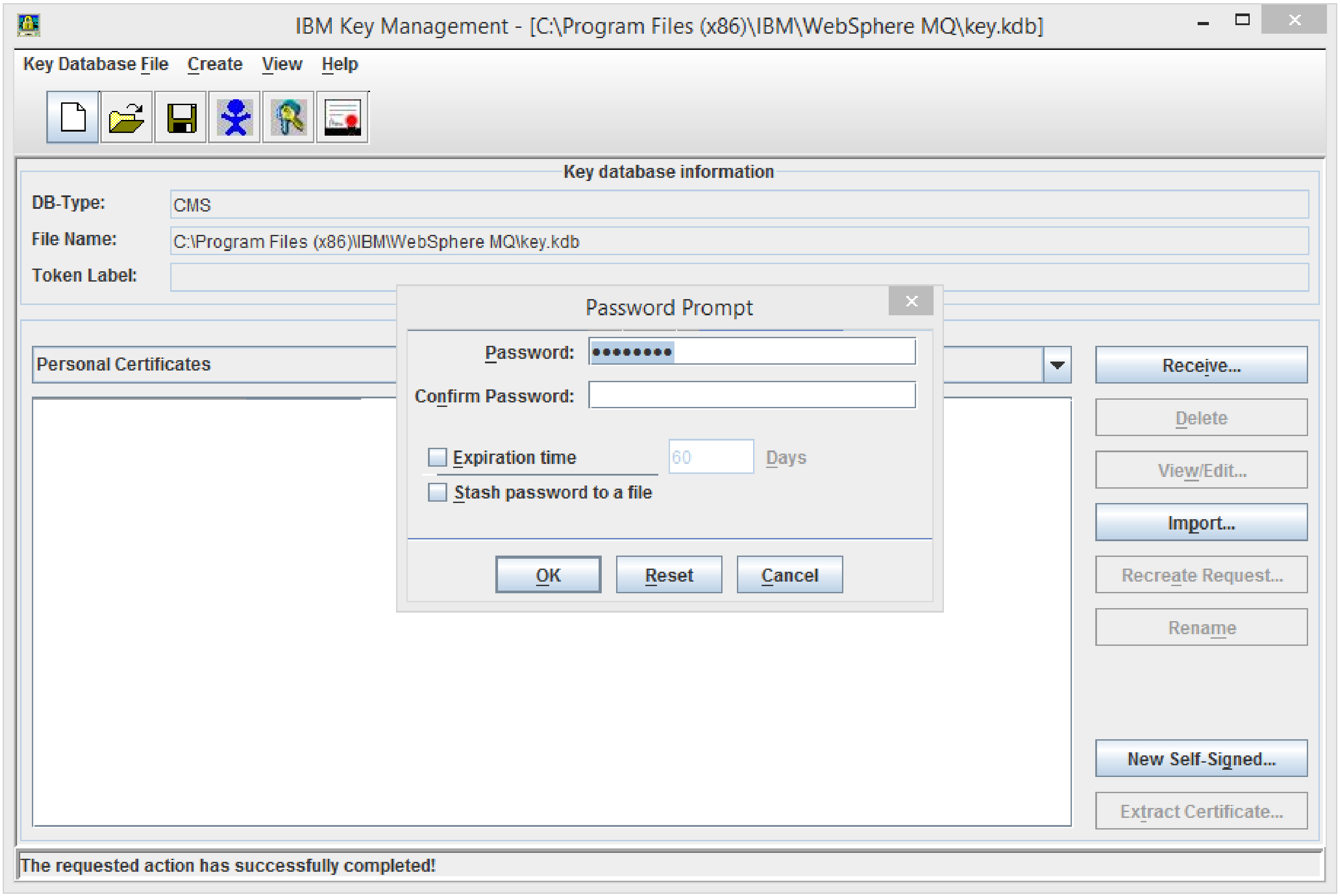
Key database type: CMS

File Name: key.kdb

Location: C:\Program Files (x86)\IBM\WebSphere MQ\

The requested action has successfully completed!

iKeyMan New Keystore -2



IBM Key Management - [C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb]

Key database information

DB-Type: CMS

File Name: C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb

Token Label:

Password Prompt

Password: ●●●●●●●●●

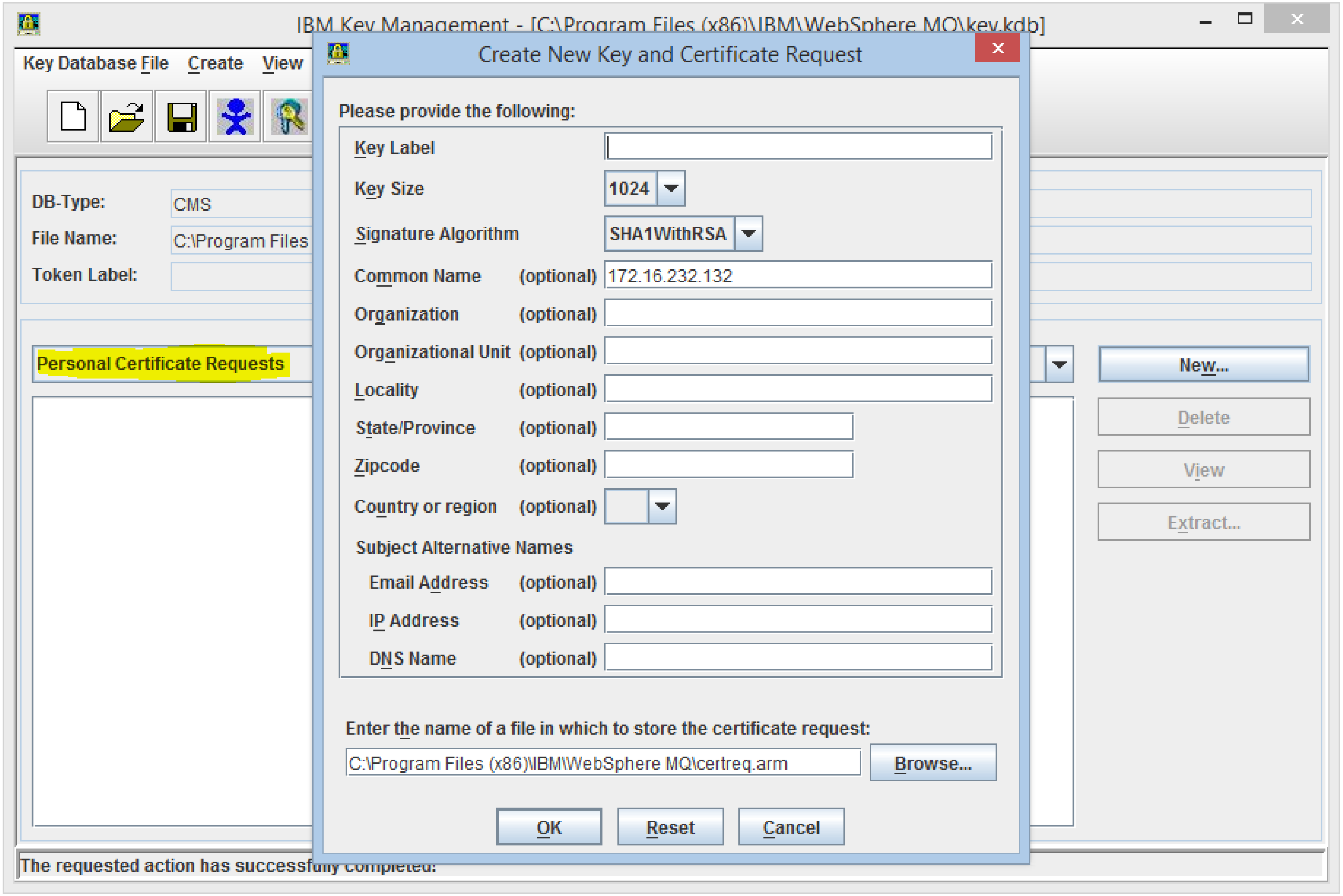
Confirm Password:

☐ Expiration time 60 Days

☐ Stash password to a file

The requested action has successfully completed!

iKeyMan Certificate Signing Request


IBM Key Management - [C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb]
Key Database File
Create
View
New...
Delete
View
Extract...
Create New Key and Certificate Request
Browse...
OK
Reset
Cancel

DB-Type: CMS
File Name: C:\Program Files
Token Label:

Personal Certificate Requests

Please provide the following:

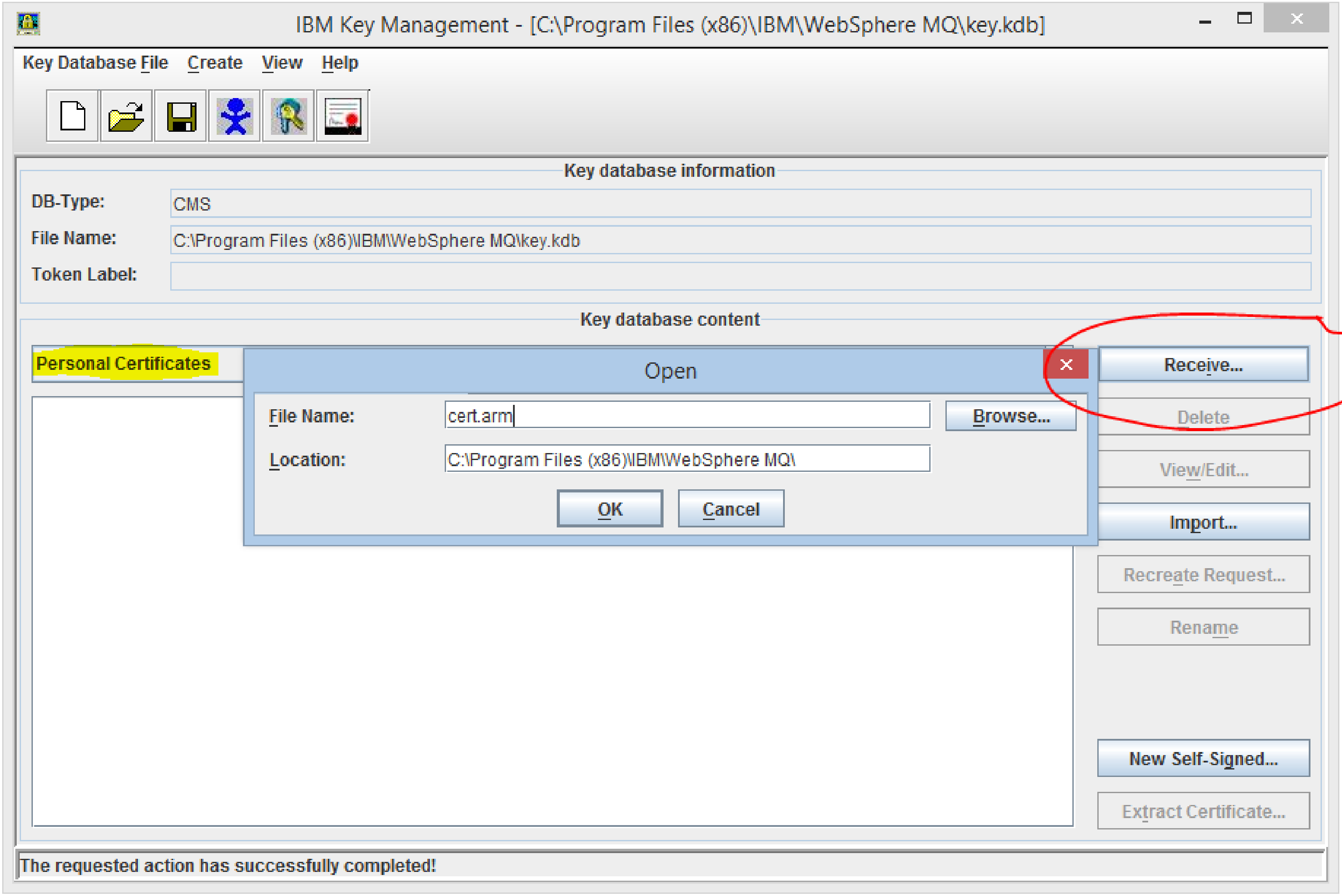
Key Label		
Key Size		1024
Signature Algorithm		SHA1WithRSA
Common Name	(optional)	172.16.232.132
Organization	(optional)	
Organizational Unit	(optional)	
Locality	(optional)	
State/Province	(optional)	
Zipcode	(optional)	
Country or region	(optional)	
Subject Alternative Names		
Email Address	(optional)	
IP Address	(optional)	
DNS Name	(optional)	

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

C:\Program Files (x86)\IBM\WebSphere MQ\certreq.arm

The requested action has successfully completed.

iKeyMan Receiving Response from CA

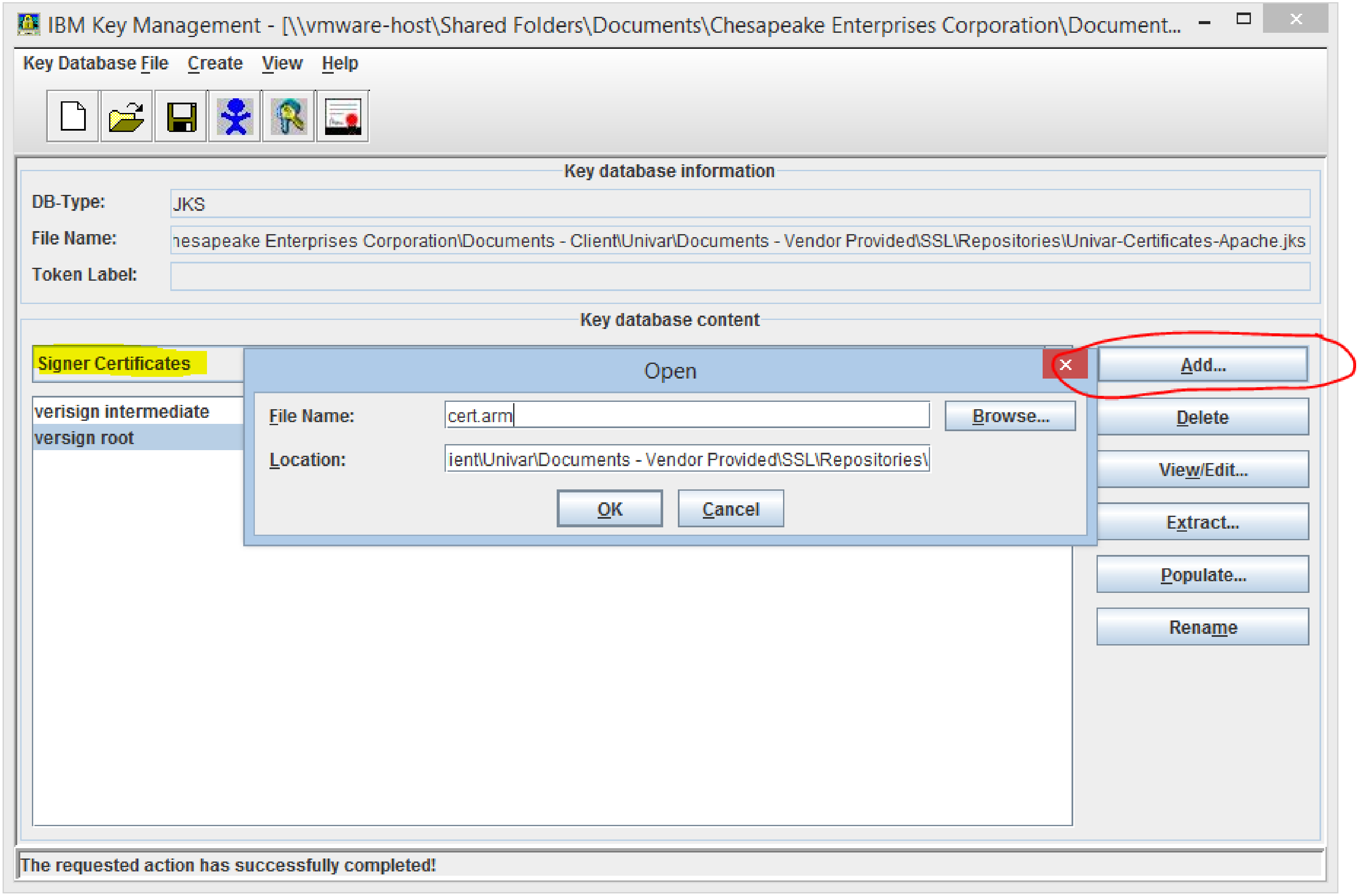


DB-Type: CMS
File Name: C:\Program Files (x86)\IBM\WebSphere MQ\key.kdb
Token Label:

Open
File Name: cert.arm
Location: C:\Program Files (x86)\IBM\WebSphere MQ\

The requested action has successfully completed!

iKeyMan Adding Signer Certificates



DB-Type: JKS

File Name: hesapeake Enterprises Corporation\Documents - Client\Univar\Documents - Vendor Provided\SSL\Repositories\Univar-Certificates-Apache.jks

Token Label:

verisign intermediate

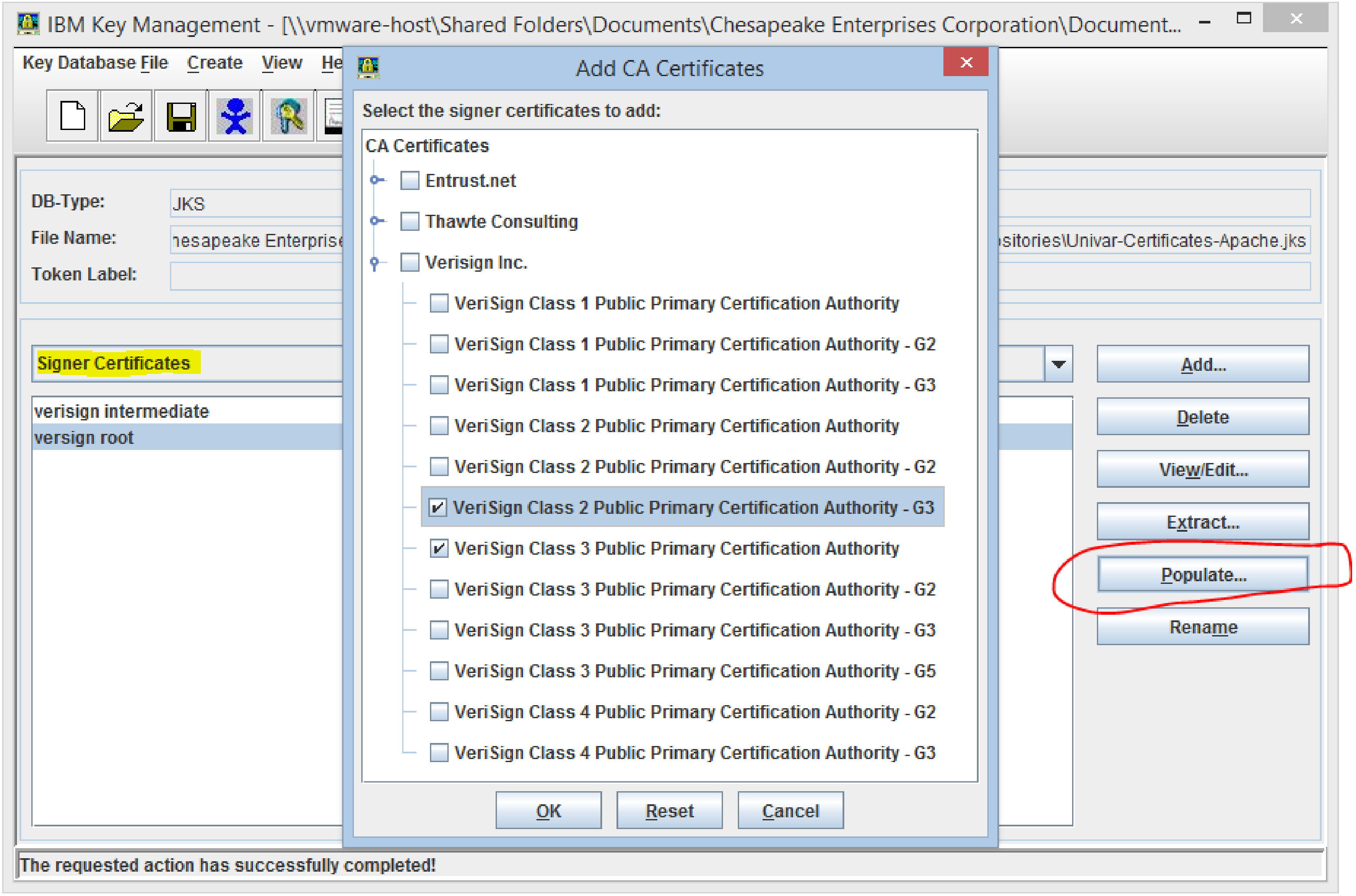
versign root

File Name: cert.arm

Location: ient\Univar\Documents - Vendor Provided\SSL\Repositories\

The requested action has successfully completed!

iKeyMan Populating Signer Certificates



IBM Key Management - [\\vmware-host\Shared Folders\Documents\Chesapeake Enterprises Corporation\Document...

Select the signer certificates to add:

CA Certificates
- [ ] Entrust.net
- [ ] Thawte Consulting
- [ ] Verisign Inc.
  - [ ] VeriSign Class 1 Public Primary Certification Authority
  - [ ] VeriSign Class 1 Public Primary Certification Authority - G2
  - [ ] VeriSign Class 1 Public Primary Certification Authority - G3
  - [ ] VeriSign Class 2 Public Primary Certification Authority
  - [ ] VeriSign Class 2 Public Primary Certification Authority - G2
  - [✓] VeriSign Class 2 Public Primary Certification Authority - G3
  - [✓] VeriSign Class 3 Public Primary Certification Authority
  - [ ] VeriSign Class 3 Public Primary Certification Authority - G2
  - [ ] VeriSign Class 3 Public Primary Certification Authority - G3
  - [ ] VeriSign Class 3 Public Primary Certification Authority - G5
  - [ ] VeriSign Class 4 Public Primary Certification Authority - G2
  - [ ] VeriSign Class 4 Public Primary Certification Authority - G3

DB-Type: JKS
File Name: hesapeake Enterprise ... sitories\Univar-Certificates-Apache.jks
Token Label:

Signer Certificates

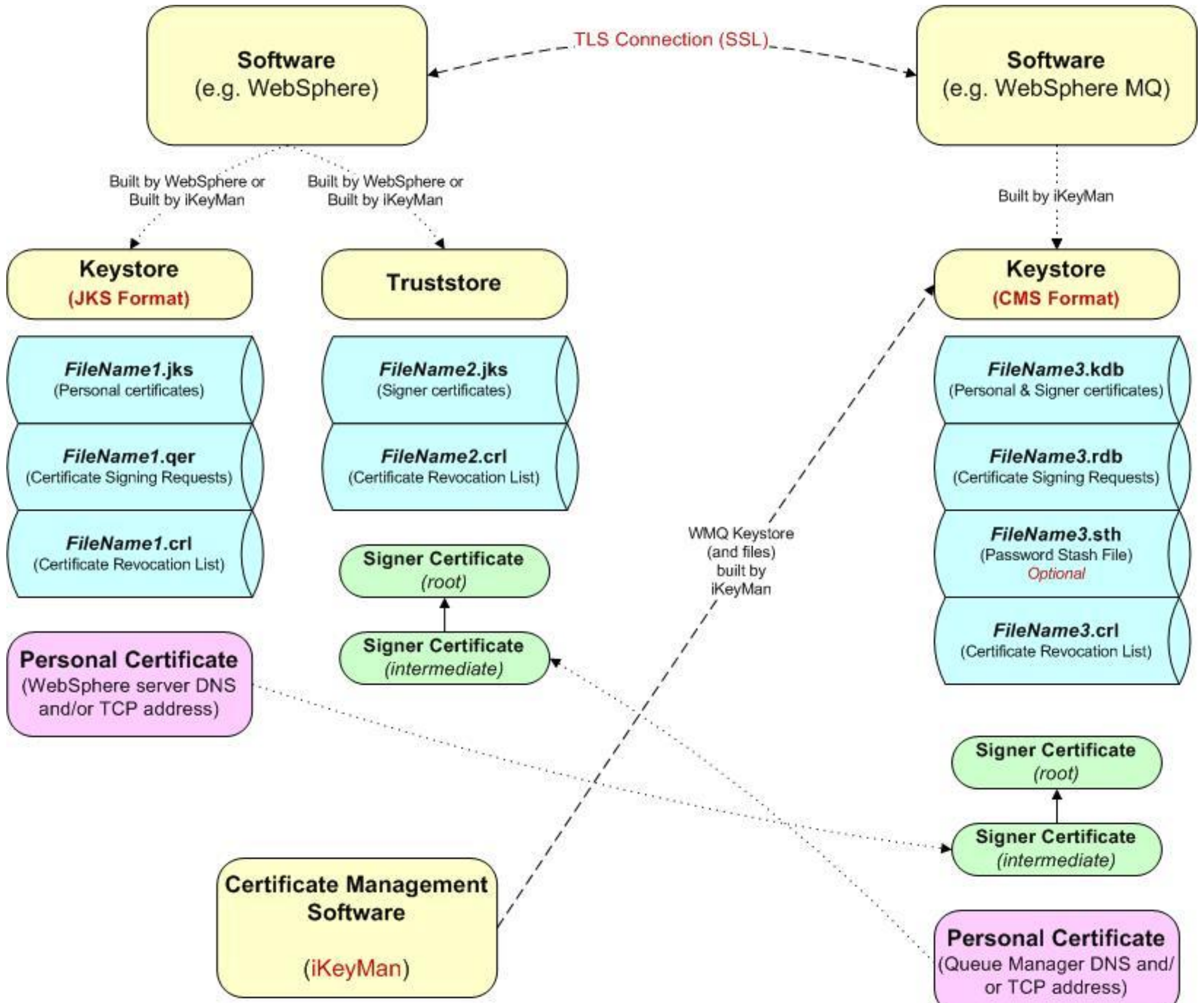
verisign intermediate
versign root

The requested action has successfully completed!

Keystore Ready to Deploy!

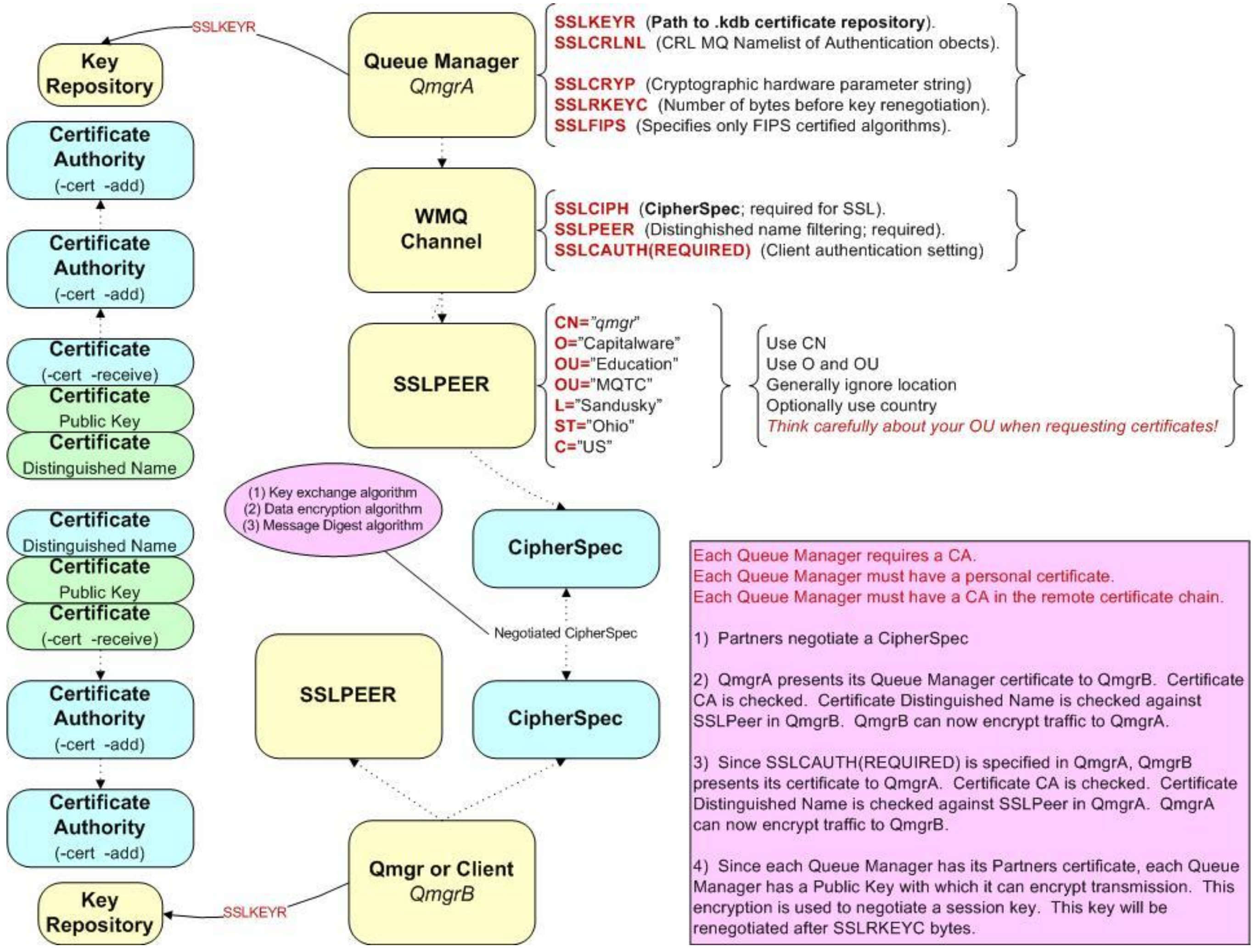
Putting it All Together

SSL/TLS Components & Connections



WMQ Channel Processing

N
O
T
E
S



SSLKEYR
Key Repository
Queue Manager
QmgrA
Certificate Authority (-cert -add)
Certificate Authority (-cert -add)
WMQ Channel
Certificate (-cert -receive)
Certificate Public Key
Certificate Distinguished Name
SSLPEER
(1) Key exchange algorithm
(2) Data encryption algorithm
(3) Message Digest algorithm
CipherSpec
Certificate Distinguished Name
Certificate Public Key
Certificate (-cert -receive)
Negotiated CipherSpec
Certificate Authority (-cert -add)
SSLPEER
CipherSpec
Certificate Authority (-cert -add)
Qmgr or Client
QmgrB
Key Repository
SSLKEYR

SSLKEYR (**Path to .kdb certificate repository**).
SSLCRLNL (CRL MQ Namelist of Authentication obects).

SSLCRYP (Cryptographic hardware parameter string)
SSLRKEYC (Number of bytes before key renegotiation).
SSLFIPS (Specifies only FIPS certified algorithms).

SSLCIPH (**CipherSpec**; required for SSL).
SSLPEER (Distinghished name filtering; required).
SSLCAUTH(REQUIRED) (Client authentication setting)

CN=*"qmgr"*
O="Capitalware"
OU="Education"
OU="MQTC"
L="Sandusky"
ST="Ohio"
C="US"

Use CN
Use O and OU
Generally ignore location
Optionally use country
Think carefully about your OU when requesting certificates!

Each Queue Manager requires a CA.
Each Queue Manager must have a personal certificate.
Each Queue Manager must have a CA in the remote certificate chain.

1) Partners negotiate a CipherSpec

2) QmgrA presents its Queue Manager certificate to QmgrB. Certificate CA is checked. Certificate Distinguished Name is checked against SSLPeer in QmgrB. QmgrB can now encrypt traffic to QmgrA.

3) Since SSLCAUTH(REQUIRED) is specified in QmgrA, QmgrB presents its certificate to QmgrA. Certificate CA is checked. Certificate Distinguished Name is checked against SSLPeer in QmgrA. QmgrA can now encrypt traffic to QmgrB.

4) Since each Queue Manager has its Partners certificate, each Queue Manager has a Public Key with which it can encrypt transmission. This encryption is used to negotiate a session key. This key will be renegotiated after SSLRKEYC bytes.

SSL & TLS Explained: A User Perspective

Reference

IIB SSL Components

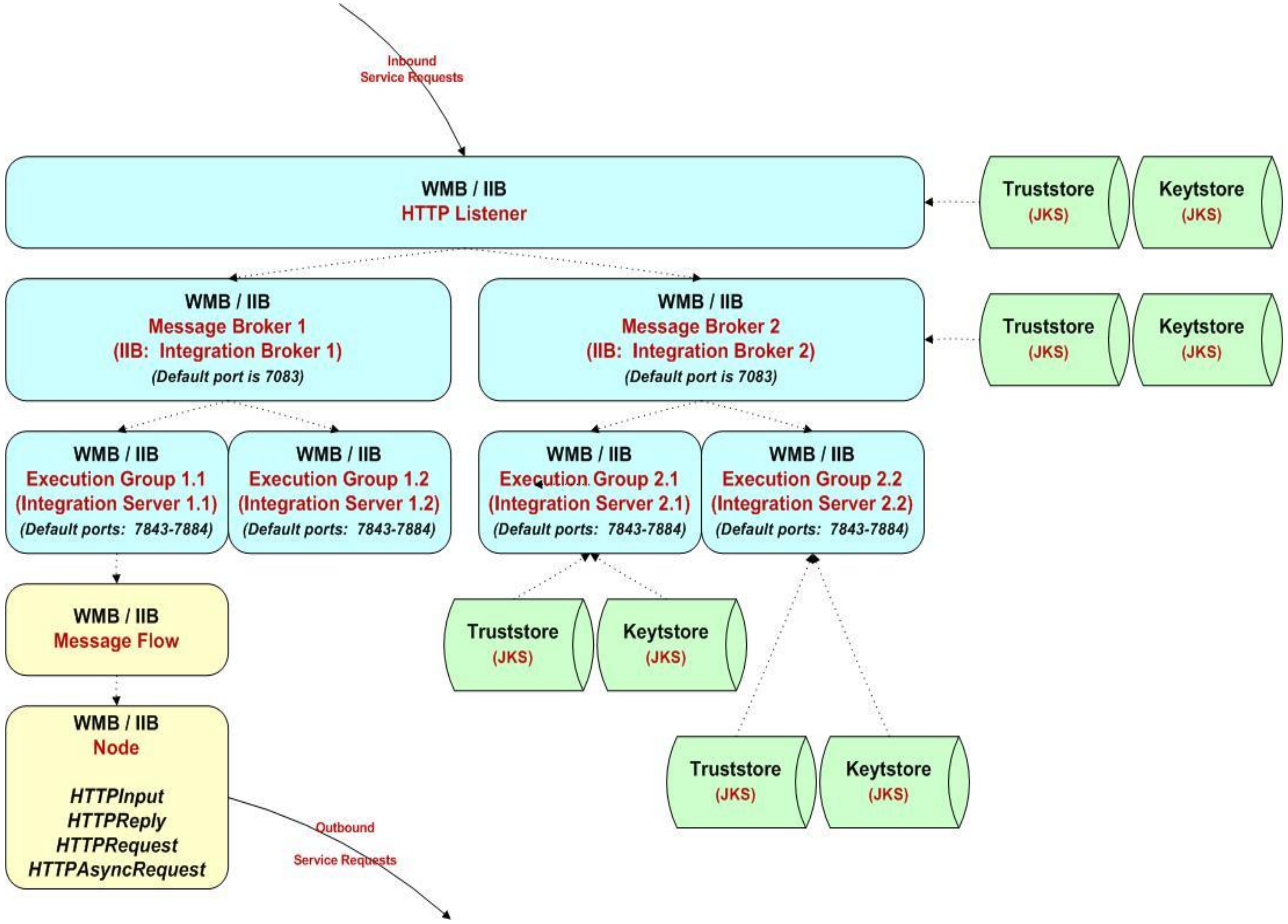
N

O

T

E

S


Inbound Service Requests
WMB / IIB HTTP Listener
Truststore (JKS)
Keytstore (JKS)
WMB / IIB Message Broker 1 (IIB: Integration Broker 1) (Default port is 7083)
WMB / IIB Message Broker 2 (IIB: Integration Broker 2) (Default port is 7083)
Truststore (JKS)
Keytstore (JKS)
WMB / IIB Execution Group 1.1 (Integration Server 1.1) (Default ports: 7843-7884)
WMB / IIB Execution Group 1.2 (Integration Server 1.2) (Default ports: 7843-7884)
WMB / IIB Execution Group 2.1 (Integration Server 2.1) (Default ports: 7843-7884)
WMB / IIB Execution Group 2.2 (Integration Server 2.2) (Default ports: 7843-7884)
WMB / IIB Message Flow
Truststore (JKS)
Keytstore (JKS)
WMB / IIB Node
HTTPInput
HTTPReply
HTTPRequest
HTTPAsyncRequest
Truststore (JKS)
Keytstore (JKS)
Outbound Service Requests

WebSphere SSL Components

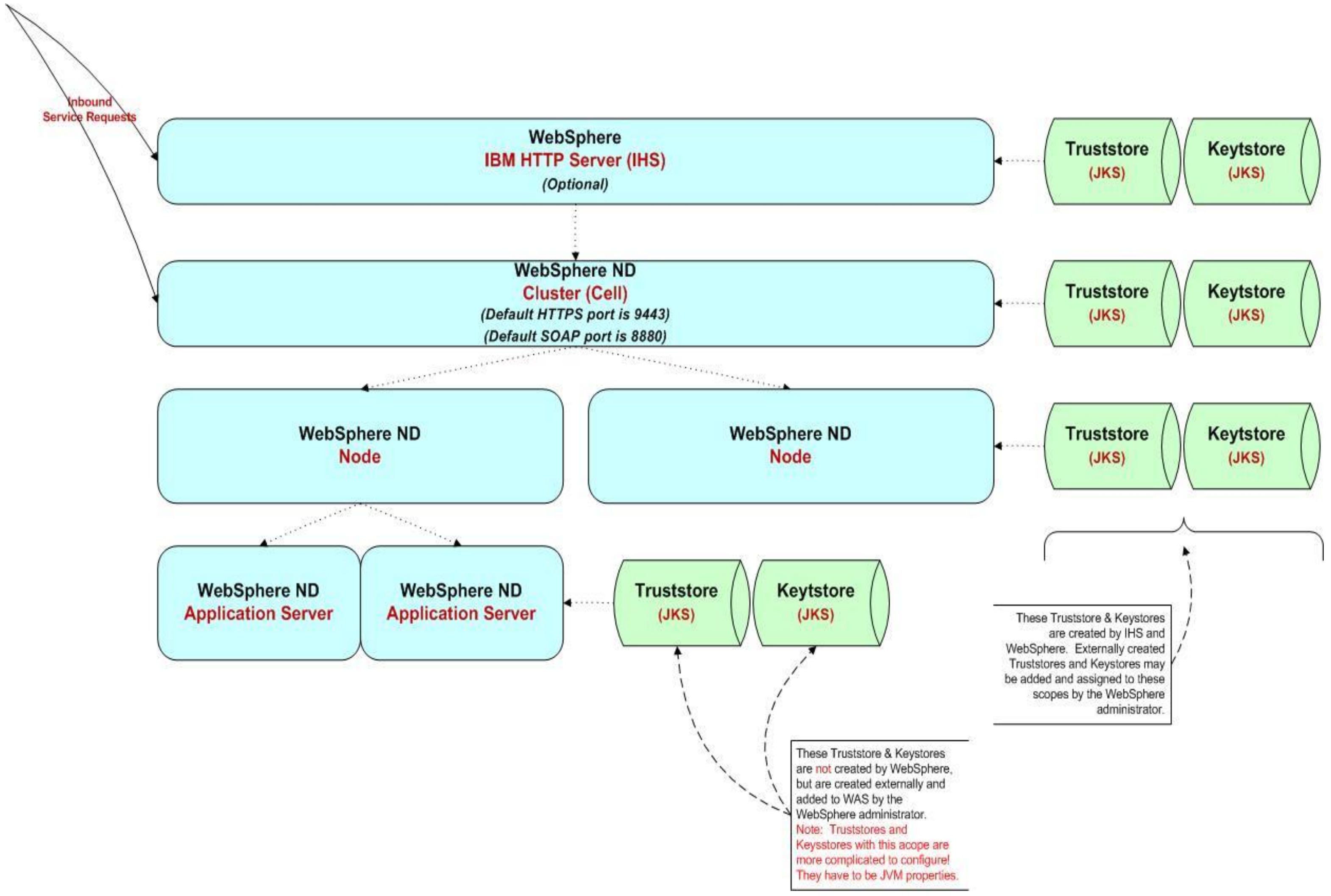
N

O

T

E

S


Inbound Service Requests
WebSphere IBM HTTP Server (IHS) (Optional)
Truststore (JKS)
Keystore (JKS)
WebSphere ND Cluster (Cell) (Default HTTPS port is 9443) (Default SOAP port is 8880)
Truststore (JKS)
Keystore (JKS)
WebSphere ND Node
WebSphere ND Node
Truststore (JKS)
Keystore (JKS)
WebSphere ND Application Server
WebSphere ND Application Server
Truststore (JKS)
Keystore (JKS)

These Truststore & Keystores are created by IHS and WebSphere. Externally created Truststores and Keystores may be added and assigned to these scopes by the WebSphere administrator.

These Truststore & Keystores are not created by WebSphere, but are created externally and added to WAS by the WebSphere administrator.
Note: Truststores and Keysstores with this acope are more complicated to configure! They have to be JVM properties.

iKeyMan Files

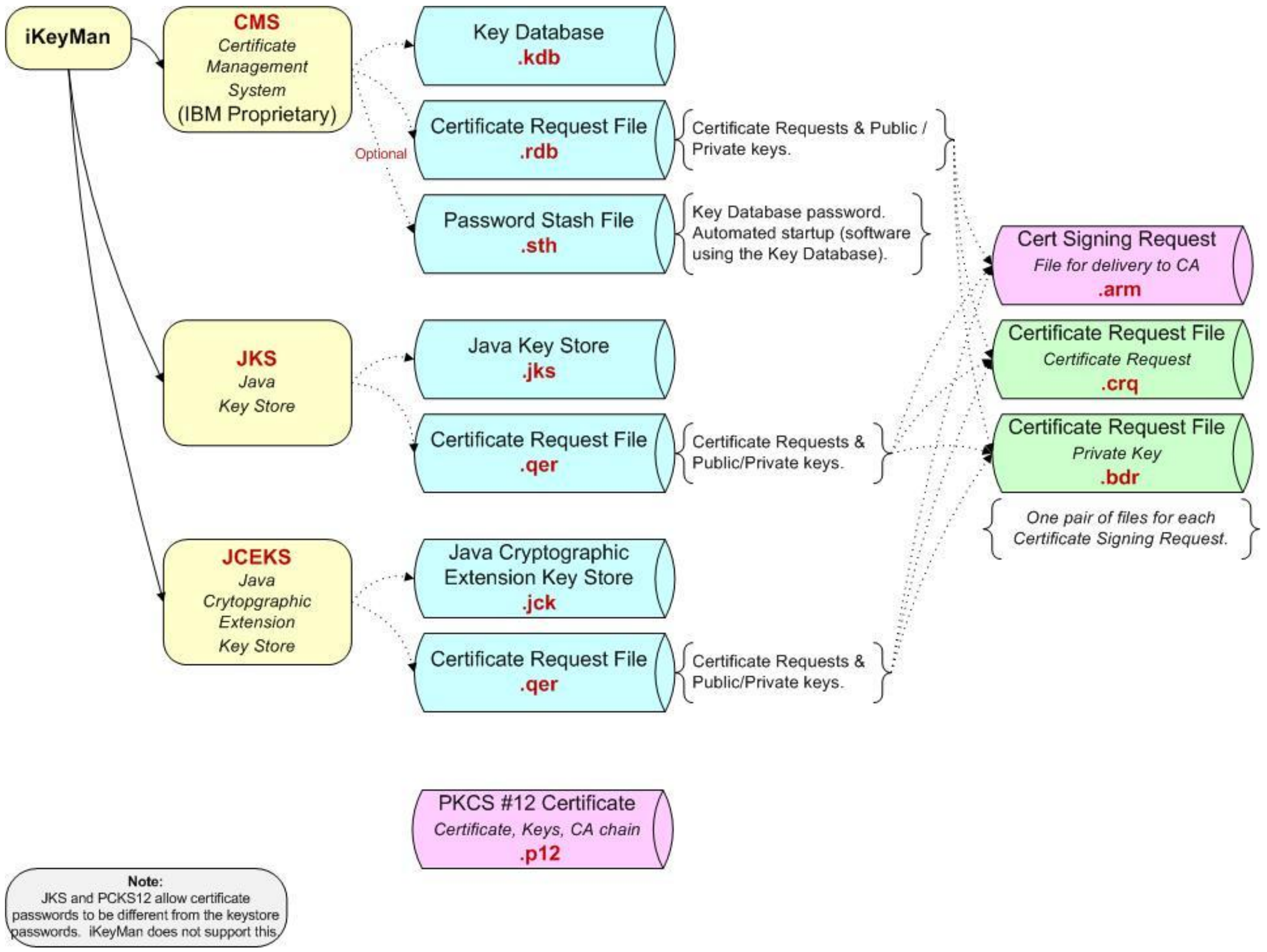
N

O

T

E

S


iKeyMan
CMS
Certificate Management System
(IBM Proprietary)
Key Database
.kdb
Certificate Request File
.rdb
Optional
Certificate Requests & Public / Private keys.
Password Stash File
.sth
Key Database password. Automated startup (software using the Key Database).
Cert Signing Request
File for delivery to CA
.arm
JKS
Java
Key Store
Java Key Store
.jks
Certificate Request File
Certificate Request
.crq
Certificate Request File
.qer
Certificate Requests & Public/Private keys.
Certificate Request File
Private Key
.bdr
One pair of files for each Certificate Signing Request.
JCEKS
Java
Crytopgraphic
Extension
Key Store
Java Cryptographic Extension Key Store
.jck
Certificate Request File
.qer
Certificate Requests & Public/Private keys.
PKCS #12 Certificate
Certificate, Keys, CA chain
.p12
Note:
JKS and PCKS12 allow certificate passwords to be different from the keystore passwords. iKeyMan does not support this.

SSL/TLS File Formats

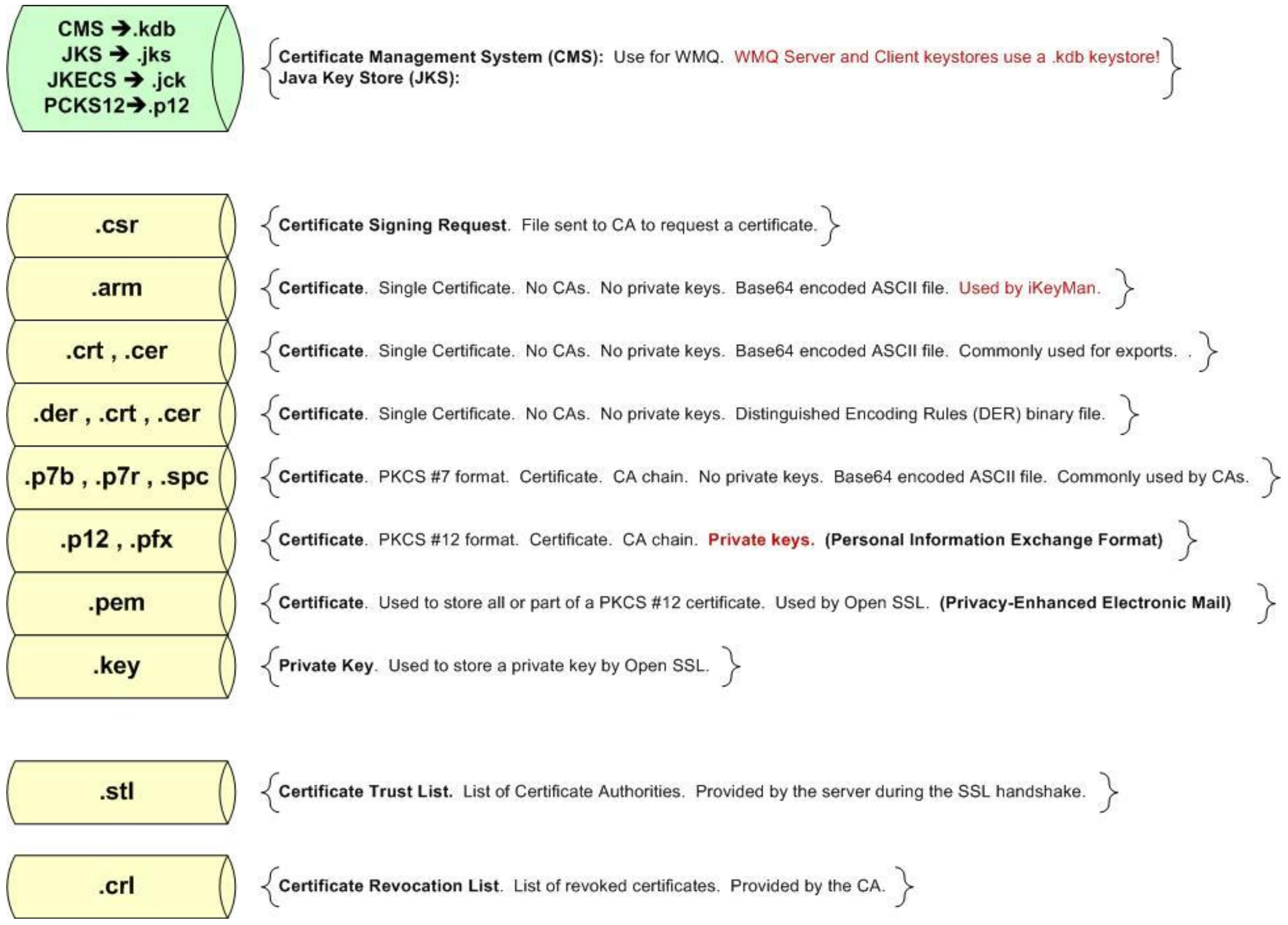
N

O

T

E

S



CMS ➔.kdb JKS ➔ .jks JKECS ➔ .jck PCKS12➔.p12	**Certificate Management System (CMS):** Use for WMQ. WMQ Server and Client keystores use a .kdb keystore! **Java Key Store (JKS):**
.csr	**Certificate Signing Request**. File sent to CA to request a certificate.
.arm	**Certificate**. Single Certificate. No CAs. No private keys. Base64 encoded ASCII file. Used by iKeyMan.
.crt , .cer	**Certificate**. Single Certificate. No CAs. No private keys. Base64 encoded ASCII file. Commonly used for exports. .
.der , .crt , .cer	**Certificate**. Single Certificate. No CAs. No private keys. Distinguished Encoding Rules (DER) binary file.
.p7b , .p7r , .spc	**Certificate**. PKCS #7 format. Certificate. CA chain. No private keys. Base64 encoded ASCII file. Commonly used by CAs.
.p12 , .pfx	**Certificate**. PKCS #12 format. Certificate. CA chain. **Private keys.** **(Personal Information Exchange Format)**
.pem	**Certificate**. Used to store all or part of a PKCS #12 certificate. Used by Open SSL. **(Privacy-Enhanced Electronic Mail)**
.key	**Private Key**. Used to store a private key by Open SSL.
.stl	**Certificate Trust List.** List of Certificate Authorities. Provided by the server during the SSL handshake.
.crl	**Certificate Revocation List**. List of revoked certificates. Provided by the CA.

iKeyMan Command Summary

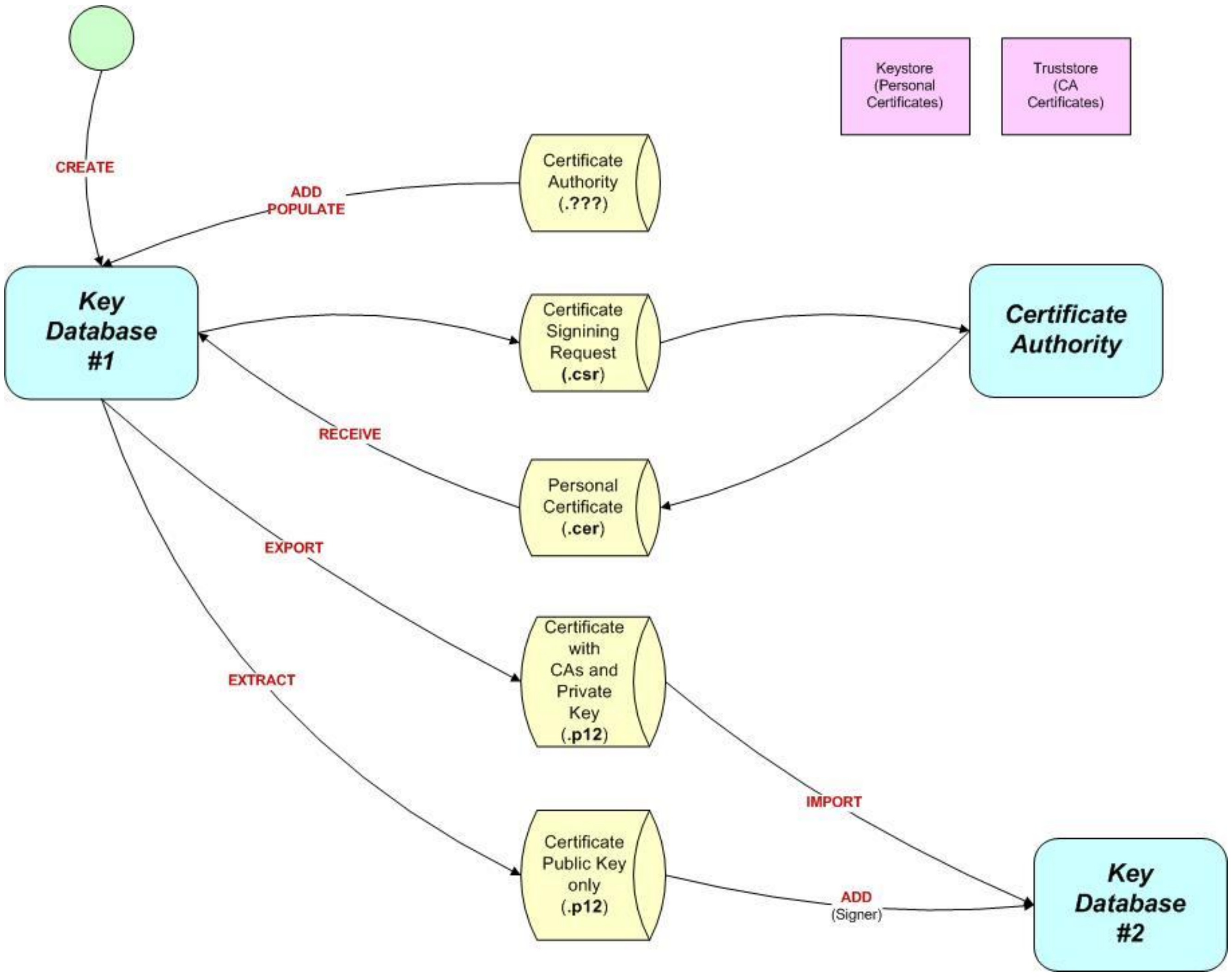
N

O

T

E

S


Keystore (Personal Certificates)
Truststore (CA Certificates)
CREATE
ADD
POPULATE
Certificate Authority (.???)
Key Database #1
Certificate Signining Request (.csr)
Certificate Authority
RECEIVE
Personal Certificate (.cer)
EXPORT
Certificate with CAs and Private Key (.p12)
EXTRACT
IMPORT
Certificate Public Key only (.p12)
ADD (Signer)
Key Database #2

iKeyMan Command Line - Keystore

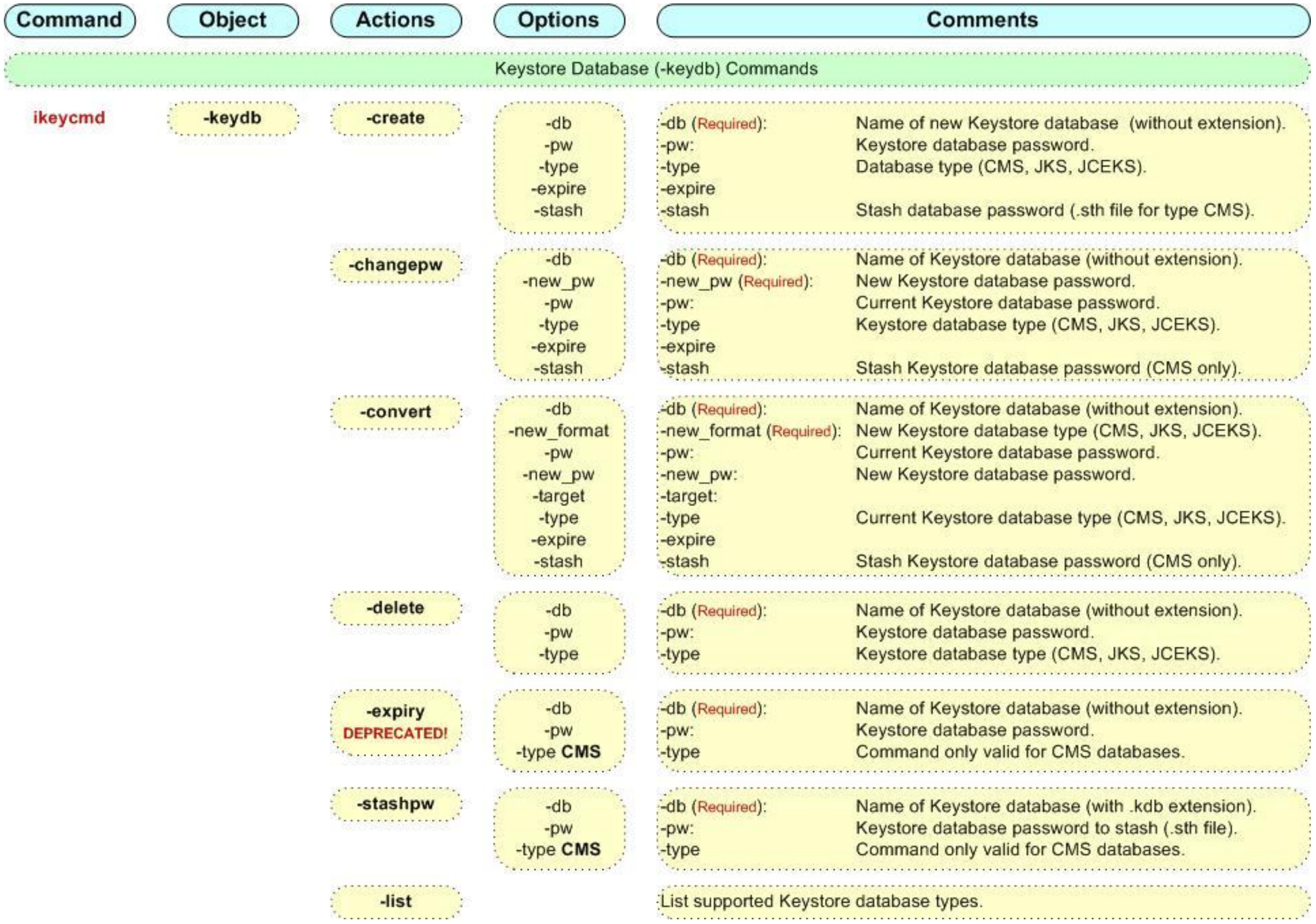
N

O

T

E

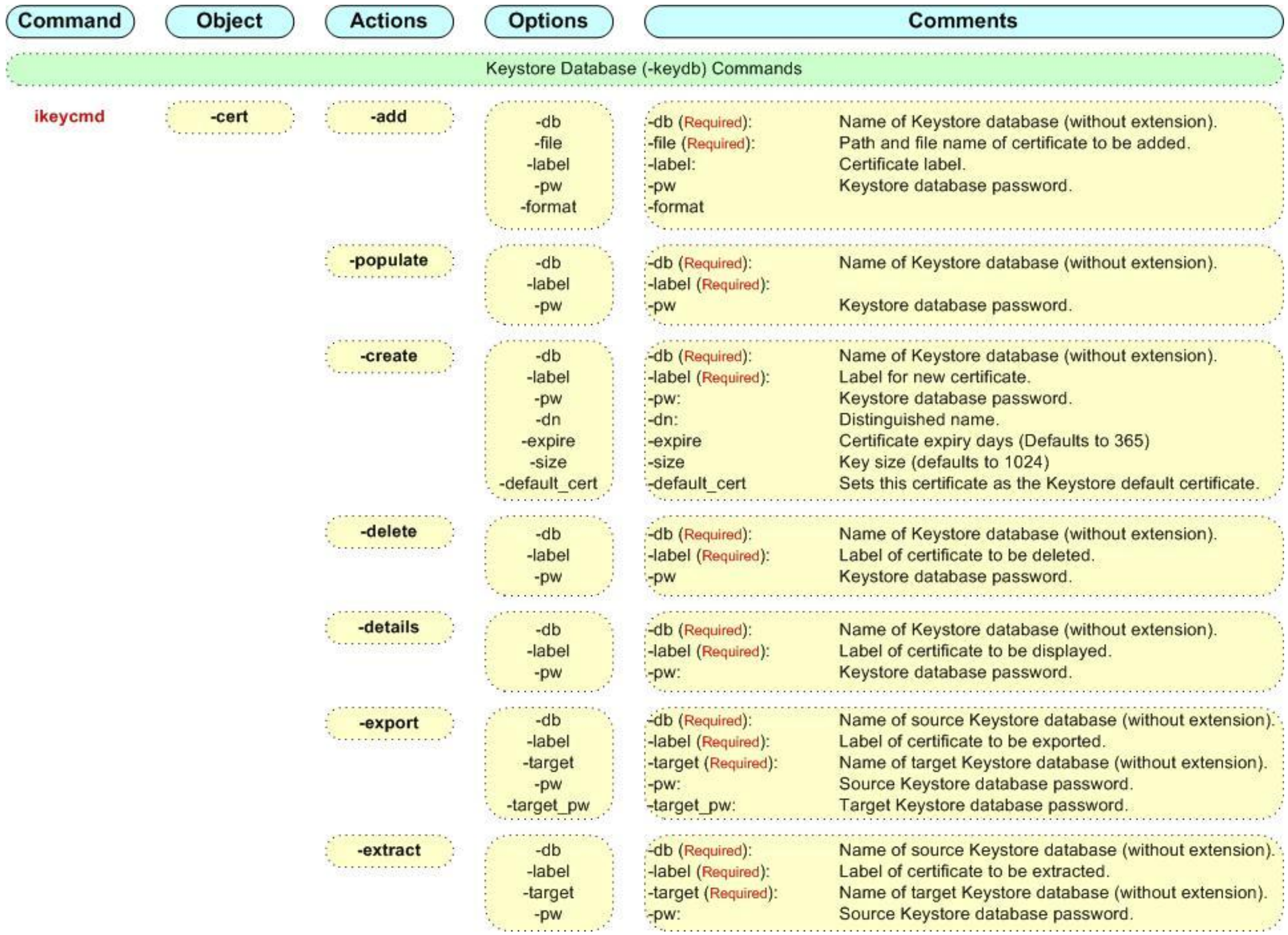
S



Command	Object	Actions	Options	Comments	
Keystore Database (-keydb) Commands					
ikeycmd	-keydb	-create	-db	-db (Required):	Name of new Keystore database (without extension).
			-pw	-pw:	Keystore database password.
			-type	-type	Database type (CMS, JKS, JCEKS).
			-expire	-expire	
			-stash	-stash	Stash database password (.sth file for type CMS).
		-changepw	-db	-db (Required):	Name of Keystore database (without extension).
			-new_pw	-new_pw (Required):	New Keystore database password.
			-pw	-pw:	Current Keystore database password.
			-type	-type	Keystore database type (CMS, JKS, JCEKS).
			-expire	-expire	
			-stash	-stash	Stash Keystore database password (CMS only).
		-convert	-db	-db (Required):	Name of Keystore database (without extension).
			-new_format	-new_format (Required):	New Keystore database type (CMS, JKS, JCEKS).
			-pw	-pw:	Current Keystore database password.
			-new_pw	-new_pw:	New Keystore database password.
			-target	-target:	
			-type	-type	Current Keystore database type (CMS, JKS, JCEKS).
			-expire	-expire	
			-stash	-stash	Stash Keystore database password (CMS only).
		-delete	-db	-db (Required):	Name of Keystore database (without extension).
			-pw	-pw:	Keystore database password.
			-type	-type	Keystore database type (CMS, JKS, JCEKS).
		-expiry DEPRECATED!	-db	-db (Required):	Name of Keystore database (without extension).
			-pw	-pw:	Keystore database password.
			-type **CMS**	-type	Command only valid for CMS databases.
		-stashpw	-db	-db (Required):	Name of Keystore database (with .kdb extension).
			-pw	-pw:	Keystore database password to stash (.sth file).
			-type **CMS**	-type	Command only valid for CMS databases.
		-list		List supported Keystore database types.	

iKeyMan Command Line – Certificates #1

N
O
T
E
S



Command	Object	Actions	Options	Comments	
Keystore Database (-keydb) Commands					
ikeycmd	-cert	-add	-db	-db (Required):	Name of Keystore database (without extension).
			-file	-file (Required):	Path and file name of certificate to be added.
			-label	-label:	Certificate label.
			-pw	-pw	Keystore database password.
			-format	-format	
		-populate	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	
			-pw	-pw	Keystore database password.
		-create	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	Label for new certificate.
			-pw	-pw:	Keystore database password.
			-dn	-dn:	Distinguished name.
			-expire	-expire	Certificate expiry days (Defaults to 365)
			-size	-size	Key size (defaults to 1024)
			-default_cert	-default_cert	Sets this certificate as the Keystore default certificate.
		-delete	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	Label of certificate to be deleted.
			-pw	-pw	Keystore database password.
		-details	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	Label of certificate to be displayed.
			-pw	-pw:	Keystore database password.
		-export	-db	-db (Required):	Name of source Keystore database (without extension).
			-label	-label (Required):	Label of certificate to be exported.
			-target	-target (Required):	Name of target Keystore database (without extension).
			-pw	-pw:	Source Keystore database password.
			-target_pw	-target_pw:	Target Keystore database password.
		-extract	-db	-db (Required):	Name of source Keystore database (without extension).
			-label	-label (Required):	Label of certificate to be extracted.
			-target	-target (Required):	Name of target Keystore database (without extension).
			-pw	-pw:	Source Keystore database password.

iKeyMan Command Line – Certificates #2

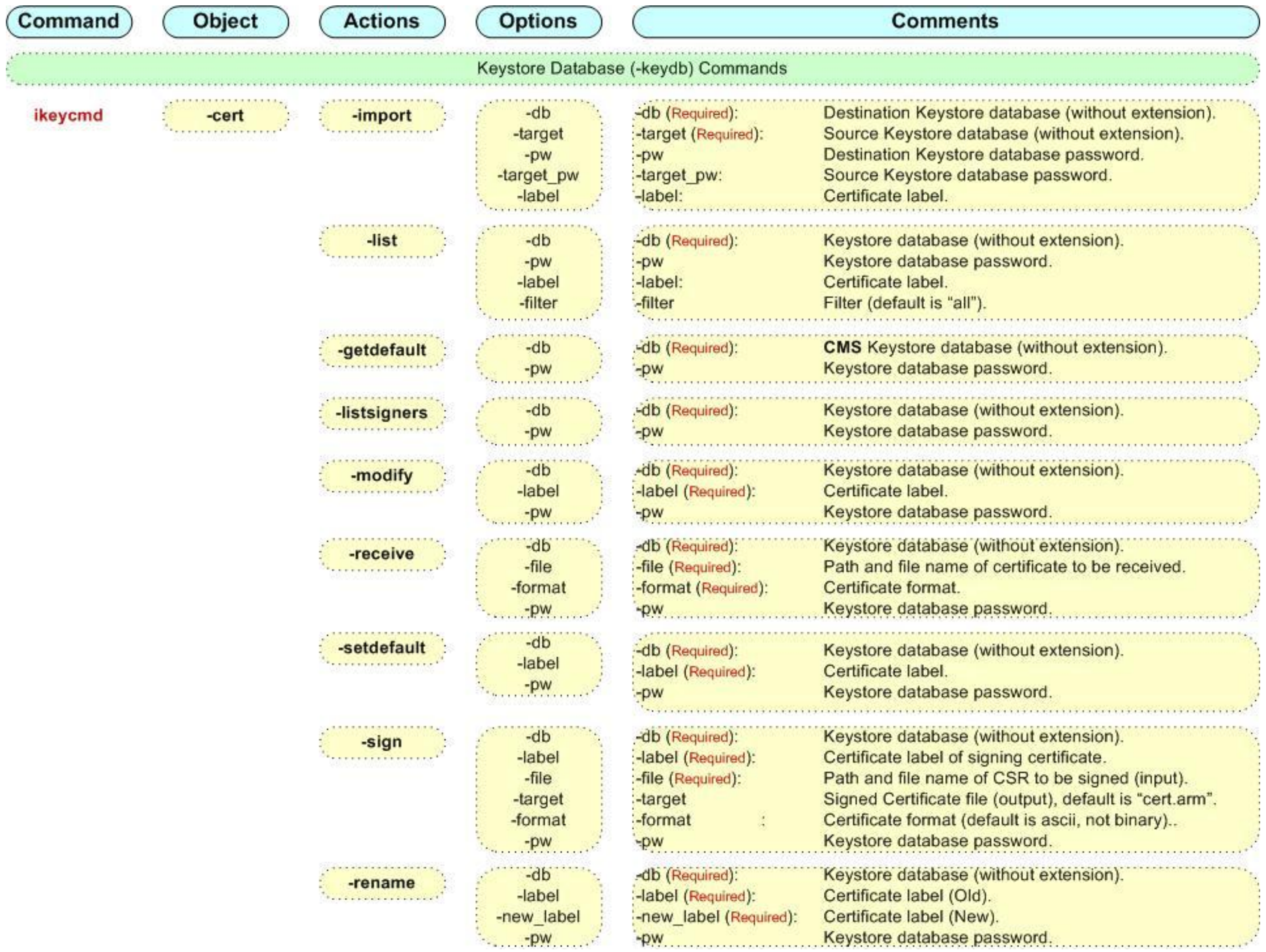
N

O

T

E

S



Command	Object	Actions	Options	Comments	
Keystore Database (-keydb) Commands					
ikeycmd	-cert	-import	-db	-db (Required):	Destination Keystore database (without extension).
			-target	-target (Required):	Source Keystore database (without extension).
			-pw	-pw	Destination Keystore database password.
			-target_pw	-target_pw:	Source Keystore database password.
			-label	-label:	Certificate label.
		-list	-db	-db (Required):	Keystore database (without extension).
			-pw	-pw	Keystore database password.
			-label	-label:	Certificate label.
			-filter	-filter	Filter (default is "all").
		-getdefault	-db	-db (Required):	**CMS** Keystore database (without extension).
			-pw	-pw	Keystore database password.
		-listsigners	-db	-db (Required):	Keystore database (without extension).
			-pw	-pw	Keystore database password.
		-modify	-db	-db (Required):	Keystore database (without extension).
			-label	-label (Required):	Certificate label.
			-pw	-pw	Keystore database password.
		-receive	-db	-db (Required):	Keystore database (without extension).
			-file	-file (Required):	Path and file name of certificate to be received.
			-format	-format (Required):	Certificate format.
			-pw	-pw	Keystore database password.
		-setdefault	-db	-db (Required):	Keystore database (without extension).
			-label	-label (Required):	Certificate label.
			-pw	-pw	Keystore database password.
		-sign	-db	-db (Required):	Keystore database (without extension).
			-label	-label (Required):	Certificate label of signing certificate.
			-file	-file (Required):	Path and file name of CSR to be signed (input).
			-target	-target	Signed Certificate file (output), default is "cert.arm".
			-format	-format :	Certificate format (default is ascii, not binary)..
			-pw	-pw	Keystore database password.
		-rename	-db	-db (Required):	Keystore database (without extension).
			-label	-label (Required):	Certificate label (Old).
			-new_label	-new_label (Required):	Certificate label (New).
			-pw	-pw	Keystore database password.

iKeyMan Command Line – Other

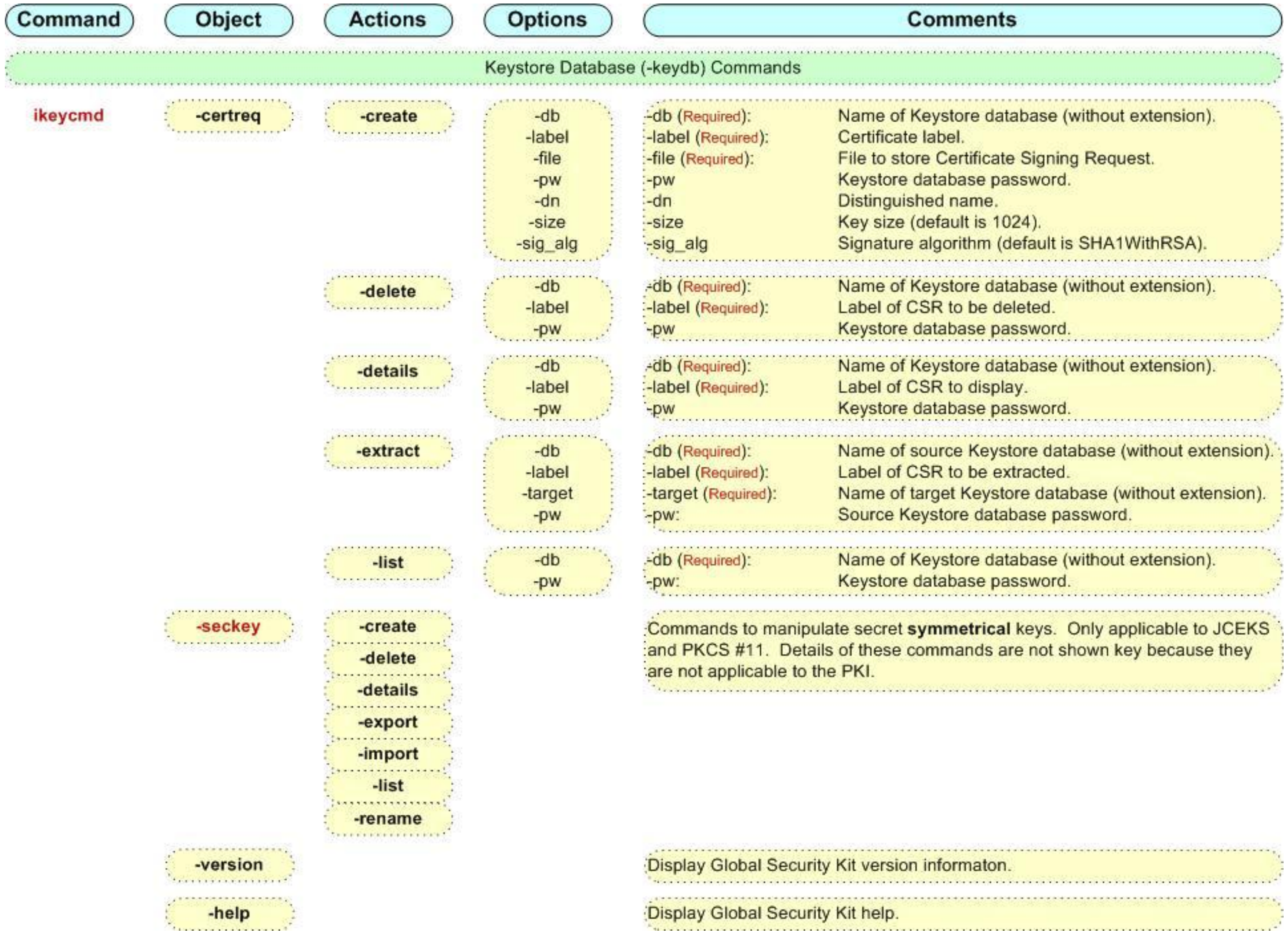
N

O

T

E

S



Command	Object	Actions	Options	Comments	
Keystore Database (-keydb) Commands					
ikeycmd	-certreq	-create	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	Certificate label.
			-file	-file (Required):	File to store Certificate Signing Request.
			-pw	-pw	Keystore database password.
			-dn	-dn	Distinguished name.
			-size	-size	Key size (default is 1024).
			-sig_alg	-sig_alg	Signature algorithm (default is SHA1WithRSA).
		-delete	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	Label of CSR to be deleted.
			-pw	-pw	Keystore database password.
		-details	-db	-db (Required):	Name of Keystore database (without extension).
			-label	-label (Required):	Label of CSR to display.
			-pw	-pw	Keystore database password.
		-extract	-db	-db (Required):	Name of source Keystore database (without extension).
			-label	-label (Required):	Label of CSR to be extracted.
			-target	-target (Required):	Name of target Keystore database (without extension).
			-pw	-pw:	Source Keystore database password.
		-list	-db	-db (Required):	Name of Keystore database (without extension).
			-pw	-pw:	Keystore database password.
	-seckey	-create		Commands to manipulate secret **symmetrical** keys. Only applicable to JCEKS and PKCS #11. Details of these commands are not shown key because they are not applicable to the PKI.	
		-delete			
		-details			
		-export			
		-import			
		-list			
		-rename			
	-version			Display Global Security Kit version informaton.	
	-help			Display Global Security Kit help.	

Reference Links

- IBM Global Security Kit v8.0 (iKeyMan)
  - IBM TechNote: AIX Installation
    - http://www-01.ibm.com/support/docview.wss?uid=swg21577384
  - IBM TechNote: Linux Installation
    - http://www-01.ibm.com/support/docview.wss?uid=swg21443726
  - IBM TechNote: WindowsInstallation
    - http://www-01.ibm.com/support/docview.wss?uid=swg21443732
- TechDoc on Global Security Kit installation
  - Contact me
- TechDoc on SSL/TLS
  - Contact me

Questions & Answers

Presenter

- Glen Brumbaugh
  - Glen.Brumbaugh@TxMQ.com
- Computer Science Background
  - Lecturer in Computer Science, University of California, Berkeley
  - Adjunct Professor in Information Systems, Golden Gate University, San Francisco
- WebSphere MQ Background (20 years plus)
  - IBM Business Enterprise Solutions Team (BEST)
    - Initial support for MQSeries v1.0
    - Trained and mentored by Hursley MQSeries staff
  - IBM U.S. Messaging Solutions Lead, GTS
  - Platforms Supported
    - MVS aka z/OS
    - UNIX (AIX, Linux, Sun OS, Sun Solaris, HP-UX)
    - Windows
    - iSeries (i5OS)
  - Programming Languages
    - C, COBOL, Java (JNI, WMQ for Java, WMQ for JMS)

SSL & TLS Explained: A User Perspective