WebSphere Application Server for z/OS

File based Keystores for WebSphere Application Server z/OS

This document can be found on the web at:
www.ibm.com/support/techdocs
Search for document number **WP101579** under the category of "White Papers"

Version Date: November 8, 2009
See "Document Change History" on page 3 for a description of the changes in this version of the document

Sridhar Talluri
WebSphere Application Server z/OS L2 Support
1-845-435-7383 (T/L: 295-7383
stalluri@us.ibm.com

Written and provided by the WebSphere Application Server for z/OS team at the
IBM Software Group, AIM

Many thanks to..

Bill O'Donnell from WebSphere Application Server Security Development.

All the **Customers** running WebSphere Application Server z/OS

Table of Contents

Overview

A Secure Sockets Layer (SSL) configuration references keystore configurations during WebSphere Application Server runtime. Keystore configurations define how the runtime for WebSphere Application Server loads and manages keystore types for Secure Sockets Layer (SSL) configurations.

z/OS keystores

WebSphere Application Server supports IBMJCE file-based keystores, Java Cryptography Extension Key Stores (JCEKS), Java Key Stores (JKS), and Public Key Cryptography Standards 12 (PKCS12), and z/OS-specific RACF (JCERACFKS) keystores. The IBMJCE file-based keystore support on z/OS is fully compatible with and similar to the support on the distributed platform. The JCERACFKS keystore uses keys and certificates that are stored and managed in RACF.

SSL communication between server and client

The server passes the signed certificate to prove its identity to the client. The client must possess the CA certificate from the same certificate authority that issued the server's certificate. The client uses the CA certificate to verify that the server's certificate is authentic. After the certificate is verified, the client can be sure that messages are truly coming from that server, not someone else.

Client setup

Personal/Signed certificates contain a private key and a public key. You can extract the public key, called the signer certificate, to a file, then import the certificate into another keystore. The client requires the signer portion of a personal certificate for Security Socket Layer (SSL) communication.

For clients, you must create a key ring and attach to it the CA certificate from the certificate authority that issued the server's certificate. For a z/OS client, you must use RACF to create a client key ring and to attach the CA certificate to that key ring.

Why file based keystore on WebSphere Application Server zOS

Below are some of the instances where you could use the filebased keystore. Please note all of the below situations can be successfully resolved by using SAF Keyring in RACF.

The objective of this article is to demonstrate an alternate way to use file based keystores instead of RACF, not to illustrate that this way is better / preferable than SAF setup.

- For any z/OS client , to establish a SSL connection with the WebSphere Application Server z/OS.
- For any WebSphere Application Server z/OS, to establish a SSL connection with the WebSphere Application Server z/OS on a remote system/lpar.
- It is often necessary to export certificates created in RACF and import them into a distributed WebSphere Application Server to prepare a distributed WebSphere Application Server for federation into an ND cell on z/OS. OR that you are using RACF as the certificate authority (CA) to issue certificates used by the distributed WebSphere Application Server cell.
- To establish a SSL connection between WebSphere Application Server on distributed platform/s with WebSphere Application Server on z/OS and vice versa.
- When a client from a previous release tries to use the addNode command to federate to a Version 6.1 deployment manager, the client must first obtain signers for a successful handshake.
- In cases where you want to avoid using OR involving RACF to create, extract and connect the certificate in RACF. Since, each z/OS client using SSL security must have a unique RACF keyring. The Certificate Authority's public certificate for all servers' it connects to using SSL must be connected to the client's keyring.

Configuring File Based Keystores in WebSphere Application Server z/OS

Here are the steps involved in exporting a RACF certificate, storing into an file-based keystore and configure the file based keystore configuration for a SSL client to make a connection using Java Secure Socket Extension.

WebSphere Application Server 6.1.x on z/OS is used to demonstrate this scenario.

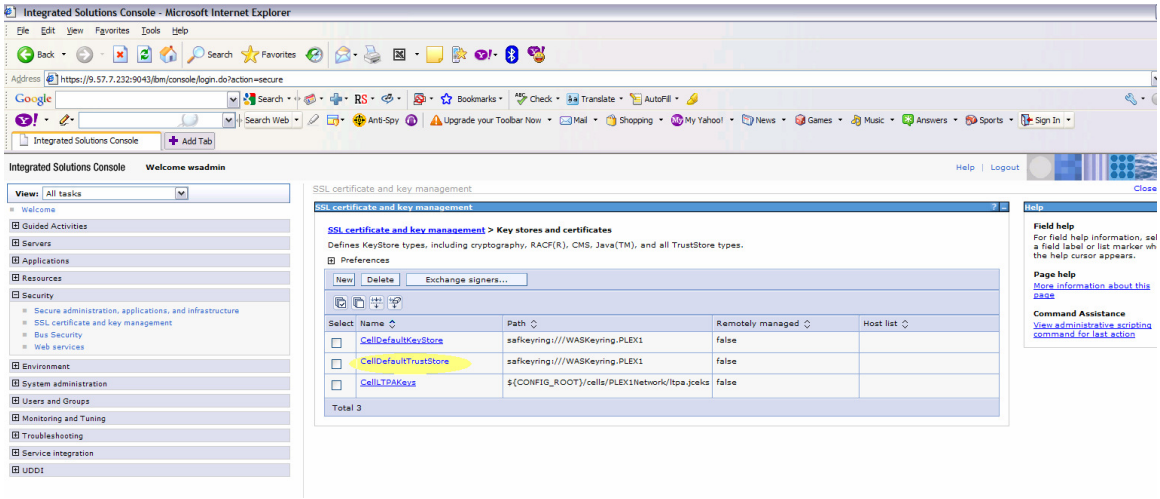
Java '**keytool**' is used to import the RACF into the keystores.

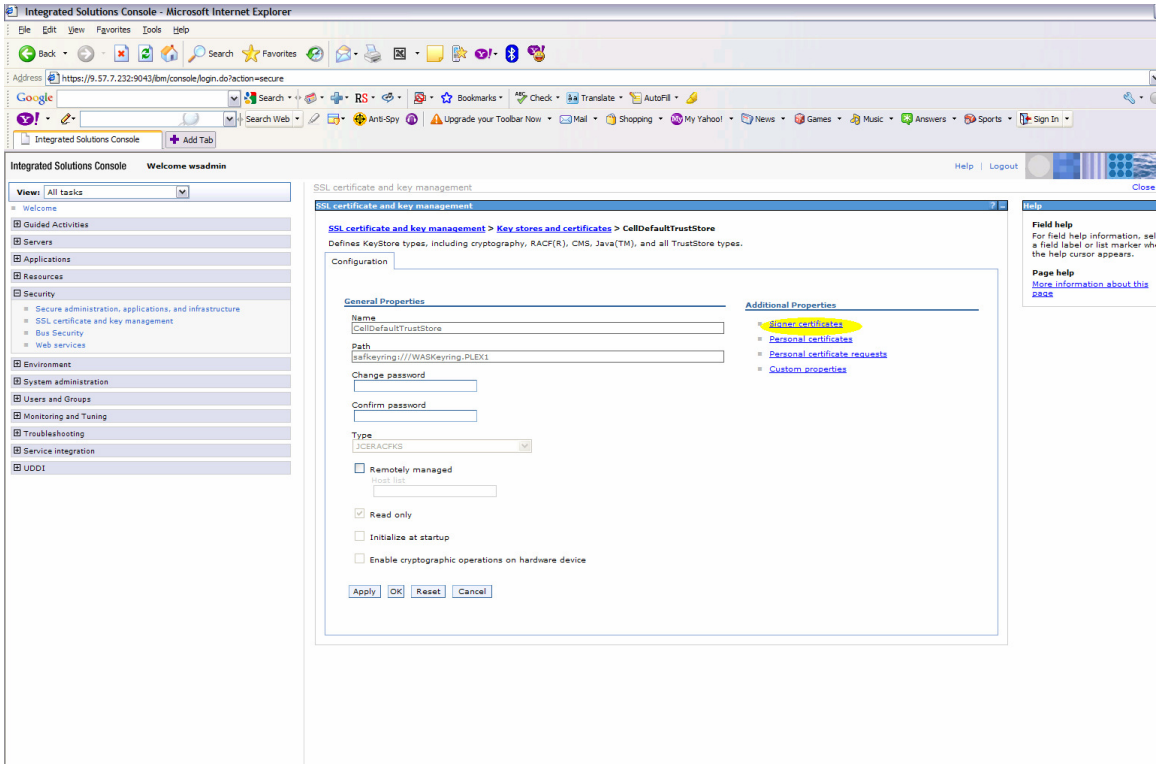
Keystore type of PKCS12 is used. Similar results can be observed when using keystore of type of JKS.

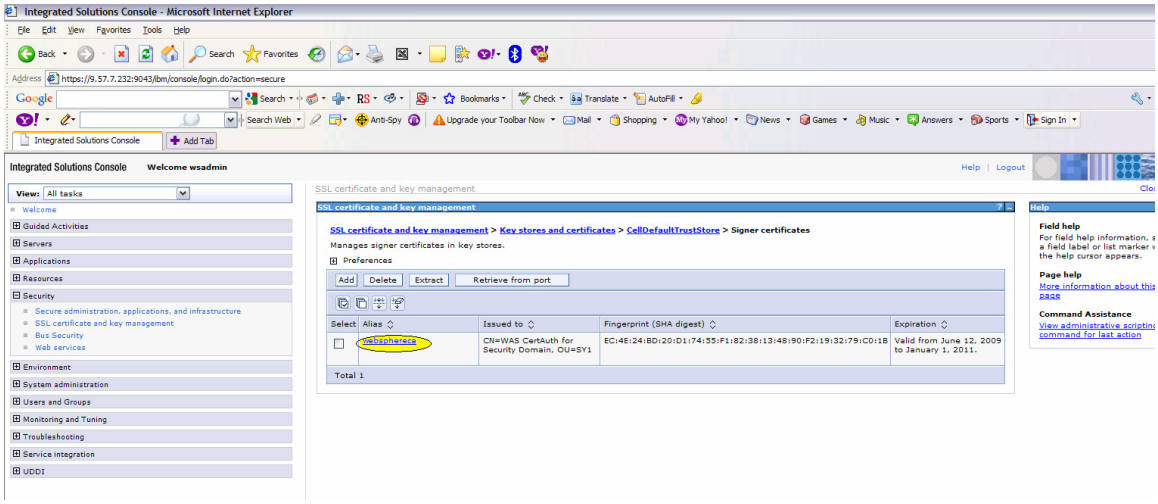
Importing RACF certificates to a file based Keystore

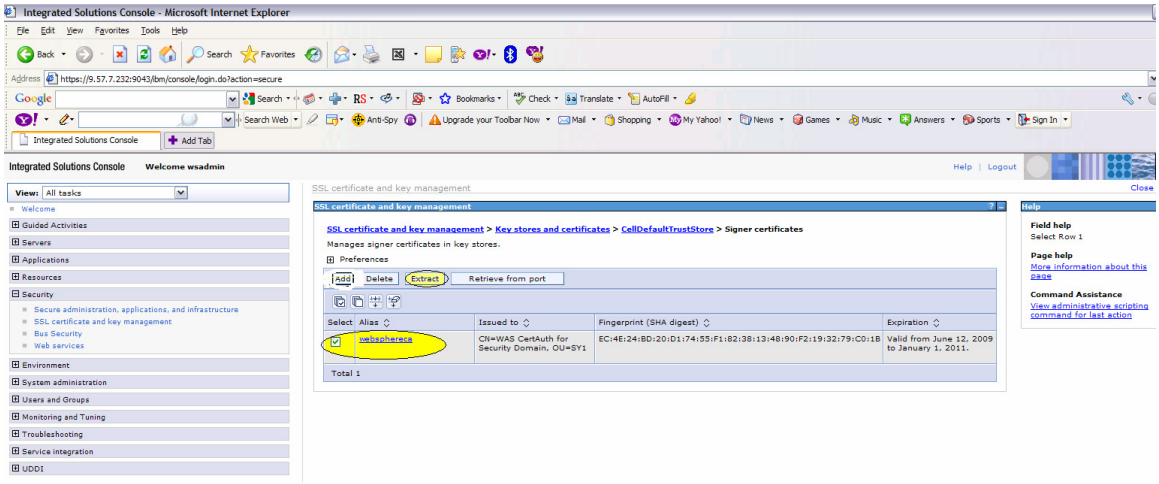
Extract the WebSphereCA cert into the HFS from the Admin Console

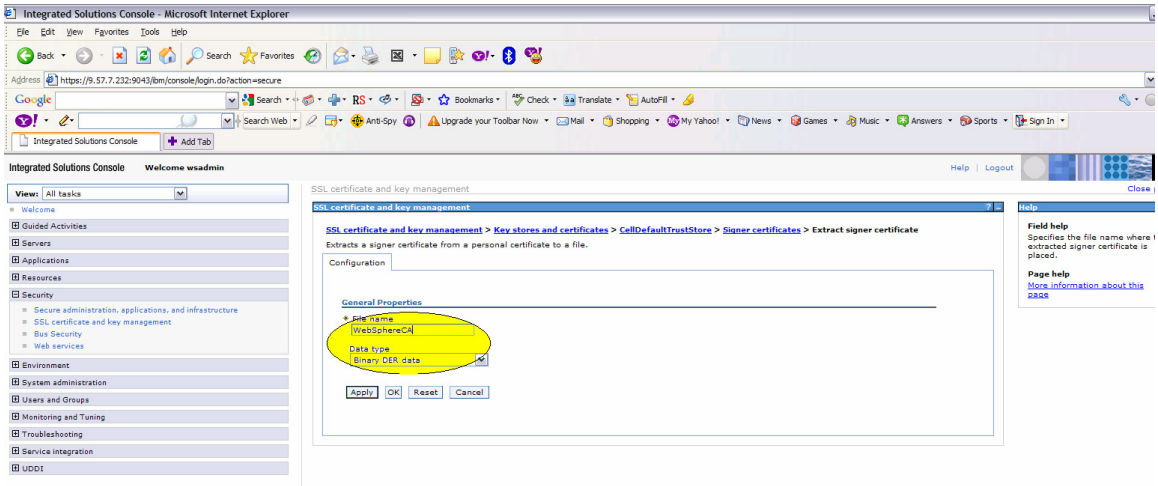
Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates > select 'websphereca'

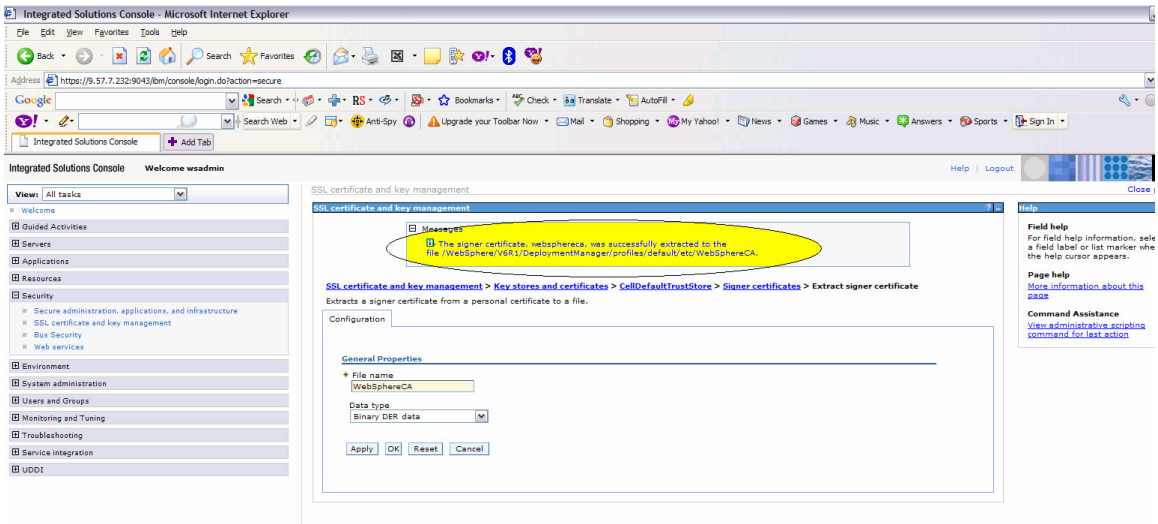



Integrated Solutions Console - Microsoft Internet Explorer
Integrated Solutions Console
Welcome wsadmin
SSL certificate and key management
SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore
Defines KeyStore types, including cryptography, RACF(R), CMS, Java(TM), and all TrustStore types.
Configuration
General Properties
Additional Properties
Signer certificates
Personal certificates
Personal certificate requests
Custom properties
Name
CellDefaultTrustStore
Path
safkeyring:///WASKeyring.PLEX1
Change password
Confirm password
Type
JCERACFKS
Remotely managed
Host list
Read only
Initialize at startup
Enable cryptographic operations on hardware device
Apply
OK
Reset
Cancel


Integrated Solutions Console - Microsoft Internet Explorer
Integrated Solutions Console
Welcome wsadmin
SSL certificate and key management
SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates
Manages signer certificates in key stores.
Preferences
Add
Delete
Extract
Retrieve from port
Select
Alias
Issued to
Fingerprint (SHA digest)
Expiration
Total 1







click '**Extract**' give path and a file name with Data type (**Binary DER data**), Apply/**OK**

The WebSphere CA cert is now extracted as a certificate file into {**/WebSphere/V6R1/DeploymentManager/profiles/default/etc**} with the given name

{before}

```
/WebSphere/V6R1/DeploymentManager/profiles/default/etc:>ls -lart
total 128
drwxrwxr-x   3 WSADMIN  CBCFG1      8192 Jun 12 11:04 ws-security
-rwxrwxr-x   1 WSADMIN  CBCFG1       727 Jun 12 11:04 serverCert.arm
-rwxrwxr-x   1 WSADMIN  CBCFG1       727 Jun 12 11:04 clientCert.arm
-rwxrwxr-x   1 WSADMIN  CBCFG1      6696 Jun 12 11:04 DummyServerTrustFile.jks
-rwxrwxr-x   1 WSADMIN  CBCFG1      2337 Jun 12 11:04 DummyServerKeyFile.jks
-rwxrwxr-x   1 WSADMIN  CBCFG1      2334 Jun 12 11:04 DummyClientKeyFile.jks
-rwxrwxr-x   1 WSADMIN  CBCFG1       834 Jun 12 11:05 trust.p12
-rwxrwxr-x   1 WSADMIN  CBCFG1      1538 Jun 12 11:05 key.p12
-rwxrwxr-x   1 WSADMIN  CBCFG1      7267 Jun 12 11:05 DummyClientTrustFile.jks
drwxrwxr-x   3 WSADMIN  CBCFG1      8192 Jun 12 11:05 .
drwxrwxr-x  17 WSADMIN  CBCFG1      8192 Oct 15 11:06 ..
```

{after}

```
/WebSphere/V6R1/DeploymentManager/profiles/default/etc:>ls -lart
total 136
drwxrwxr-x   3 WSADMIN  CBCFG1      8192 Jun 12 11:04 ws-security
-rwxrwxr-x   1 WSADMIN  CBCFG1       727 Jun 12 11:04 serverCert.arm
-rwxrwxr-x   1 WSADMIN  CBCFG1       727 Jun 12 11:04 clientCert.arm
-rwxrwxr-x   1 WSADMIN  CBCFG1      6696 Jun 12 11:04 DummyServerTrustFile.jks
-rwxrwxr-x   1 WSADMIN  CBCFG1      2337 Jun 12 11:04 DummyServerKeyFile.jks
-rwxrwxr-x   1 WSADMIN  CBCFG1      2334 Jun 12 11:04 DummyClientKeyFile.jks
-rwxrwxr-x   1 WSADMIN  CBCFG1       834 Jun 12 11:05 trust.p12
-rwxrwxr-x   1 WSADMIN  CBCFG1      1538 Jun 12 11:05 key.p12
-rwxrwxr-x   1 WSADMIN  CBCFG1      7267 Jun 12 11:05 DummyClientTrustFile.jks
drwxrwxr-x  17 WSADMIN  CBCFG1      8192 Oct 15 11:06 ..
-rw-rw----   1 DMSR1    CBCFG1       625 Oct 15 11:53 WebSphereCA
drwxrwxr-x   3 WSADMIN  CBCFG1      8192 Oct 15 11:54 .
```

Setup z/OS client environment

Under Telnet / USS, Set the $PATH to access the WebSphere and Java binaries:

```
export PATH=$PATH:/WebSphere/V6R1/DeploymentManager/bin:.
export PATH=$PATH:/WebSphere/V6R1/DeploymentManager/java/bin:.
```

Add the CA to the file-based PKCS12 type truststore using java the keytool utility

```
/WebSphere/V6R1/DeploymentManager/profiles/default/etc:>keytool -import
   -file WebsphereCA -keystore trust.p12 -storetype PKCS12 -storepass WebAS
Owner: CN=WAS CertAuth for Security Domain, OU=SY1
Issuer: CN=WAS CertAuth for Security Domain, OU=SY1
Serial number: 0
Valid from: 6/12/09 1:00 AM until: 12/31/10 11:59 PM
Certificate fingerprints:
      MD5:  40:EF:C7:6F:36:47:47:4B:BD:8F:CE:21:67:DA:DD:F5
      SHA1: EC:4E:24:BD:20:D1:74:55:F1:82:38:13:48:90:F2:19:32:79:C0:1B
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

List and verify the CA cert just added to the truststore

```
/WebSphere/V6R1/DeploymentManager/profiles/default/etc:>keytool -list
                    -keystore trust.p12 -storetype PKCS12 -storepass WebAS
```

```
Keystore type: PKCS12

Keystore provider: IBMJCE

Your keystore contains 2 entries

cn=was certauth for security domain, ou=sy1, Dec 31, 1969,
trustedCertEntry,

Certificate fingerprint (MD5):
40:EF:C7:6F:36:47:47:4B:BD:8F:CE:21:67:DA:DD:F5

default_signer, Dec 31, 1969, trustedCertEntry,

Certificate fingerprint (MD5):
4B:49:9B:8D:17:99:3D:2D:A2:D2:54:D1:8E:0C:43:1E
```

Configure the z/OS Client to use the file based keystores

Make the below changes to the **ssl.client.props** to point to the **key.p12**, **trust.p12** as the keystore/s

```
/WebSphere/V6R1/DeploymentManager/profiles/default/properties/ssl.client.props
# KeyStore information
com.ibm.ssl.keyStoreName=ClientDefaultKeyStore
#com.ibm.ssl.keyStore=safkeyring:///WASKeyring.PLEX1
#com.ibm.ssl.keyStorePassword={xor}Lz4sLCgwLTs=
#com.ibm.ssl.keyStoreType=JCERACFKS
#com.ibm.ssl.keyStoreProvider=IBMJCE
#com.ibm.ssl.keyStoreFileBased=false
com.ibm.ssl.keyStore=/WebSphere/V6R1/DeploymentManager/profiles/default/etc/key.p12
com.ibm.ssl.keyStorePassword=WebAS
com.ibm.ssl.keyStoreType=PKCS12
com.ibm.ssl.keyStoreProvider=IBMJCE
com.ibm.ssl.keyStoreFileBased=true
 :
# Truststore information
com.ibm.ssl.trustStoreName=ClientDefaultTrustStore
#com.ibm.ssl.trustStore=safkeyring:///WASKeyring.PLEX1
#com.ibm.ssl.trustStorePassword={xor}Lz4sLCgwLTs=
#com.ibm.ssl.trustStoreType=JCERACFKS
#com.ibm.ssl.trustStoreProvider=IBMJCE
#com.ibm.ssl.trustStoreFileBased=false
com.ibm.ssl.trustStore=/WebSphere/V6R1/DeploymentManager/profiles/default/etc/trust.p12
com.ibm.ssl.trustStorePassword=WebAS
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
```

Password encoding using PropFilePasswordEncoder utility

This will create the backup and also convert to ascii:

```
/WebSphere/V6R1/DeploymentManager/profiles/default/properties:>PropFilePasswo
rdEncoder.sh ssl.client.props com.ibm.ssl.keyStorePassword

Create a backup file of the original properties file which contains unencoded
passwords? (y/n): y

NOTE:  Backup file
/WebSphere/V6R1/DeploymentManager/profiles/default/properties/ssl.client.prop
s.bak contains unencoded passwords
```

```
/WebSphere/V6R1/DeploymentManager/profiles/default/properties:>PropFilePasswo
rdEncoder.sh ssl.client.props com.ibm.ssl.trustStorePassword

Create a backup file of the original properties file which contains unencoded
passwords? (y/n): y

NOTE:  Backup file
/WebSphere/V6R1/DeploymentManager/profiles/default/properties/ssl.client.prop
s.bak contains unencoded passwords
```

Verify if ssl.client.props is in ascii / ebcdic

```
/WebSphere/V6R1/DeploymentManager/profiles/default/properties:>file
ssl.client.props*

ssl.client.props:       binary data

ssl.client.props.bak:   text
```

The password for the keystore/s in ssl.client.props is now encoded to the below.

```
com.ibm.ssl.keyStorePassword={xor}CDo9Hgw=

com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=
```

ssl.client.props will now look like

```
com.ibm.ssl.keyStore=/WebSphere/V6R1/DeploymentManager/profiles/default/etc/key.p12

com.ibm.ssl.keyStorePassword={xor}CDo9Hgw=

com.ibm.ssl.keyStoreType=PKCS12

com.ibm.ssl.keyStoreFileBased=true

...

com.ibm.ssl.trustStore=/WebSphere/V6R1/DeploymentManager/profiles/default/etc/trust.p12

com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=

com.ibm.ssl.trustStoreType=PKCS12

com.ibm.ssl.trustStoreFileBased=true
```

Using retrieveSigners utility instead of Administrative Console

> **Note:** You will have to first follow the steps described in Make the below changes to the **ssl.client.props** to point to the **key.p12**, **trust.p12** as the keystore/s

Use the below commands to list the keystores of both Client, Server.

```
/WebSphere/V6R1/DeploymentManager/profiles/default/bin:>retrieveSigners.sh
                                                        -listLocalKeyStoreNames
CWPKI0307I: The following local keystores exist on the client:
          ClientDefaultKeyStore, ClientDefaultTrustStore

/WebSphere/V6R1/DeploymentManager/profiles/default/bin:>retrieveSigners.sh
                                                       -listRemoteKeyStoreNames
CWPKI0306I: The following remote keystores exist on the specified server:
           CellDefaultKeyStore, CellLTPAKeys, LEX1Manager/DefaultIIOPSSL_key,
           SY1/DefaultIIOPSSL_trust, PLEX1Manager/DefaultIIOPSSL_trust,
           SY1/DefaultIIOPSSL_key, CellDefaultTrustStore
```

Import the SAF Keyring certificates to the file based keystore, truststore

```
/WebSphere/V6R1/DeploymentManager/profiles/default/bin:>retrieveSigners.sh
CellDefaultTrustStore ClientDefaultTrustStore -autoAcceptBootstrapSigner

CWPKI0308I: Adding signer alias "CN=BOSSXXXX.PLEX1.L2.IBM.COM, OU=PLEX1, O=IBM"
          to local keystore "ClientDefaultTrustStore" with the following SHA
          digest: EC:4E:24:BD:20:D1:74:55:F1:82:38:13:48:90:F2:19:32:79:C0:1B

CWPKI0308I: Adding signer alias "CN=WAS CertAuth for Security Domain, OU=SY1"
          to local keystore "ClientDefaultTrustStore" with the following SHA
          digest: EC:4E:24:BD:20:D1:74:55:F1:82:38:13:48:90:F2:19:32:79:C0:1B

CWPKI0309I: All signers from remote keystore already exist in local keystore.

/WebSphere/V6R1/DeploymentManager/profiles/default/bin:>retrieveSigners.sh
CellDefaultKeyStore ClientDefaultKeyStore -autoAcceptBootstrapSigners

CWPKI0308I: Adding signer alias "websphereca" to local keystore
          "ClientDefaultKeyStore" with the following SHA digest:
          EC:4E:24:BD:20:D1:74:55:F1:82:38:13:48:90:F2:19:32:79:C0:1B
```

Establish a SOAP Connection from a client using the keystores

> **Note:** The WebSphere Server cell configuration is using the SAF Keyring

```
/:>wsadmin.sh -conntype SOAP -host boss0232.plex1.l2.ibm.com -port 8879
                                                  -user ibmuser -password ibmuser

WASX7209I: Connected to process "dmgr" on node PLEX1Manager using SOAP
connector;  The type of process is: DeploymentManager

WASX7029I: For help, enter: "$Help help"wsadmin>
```

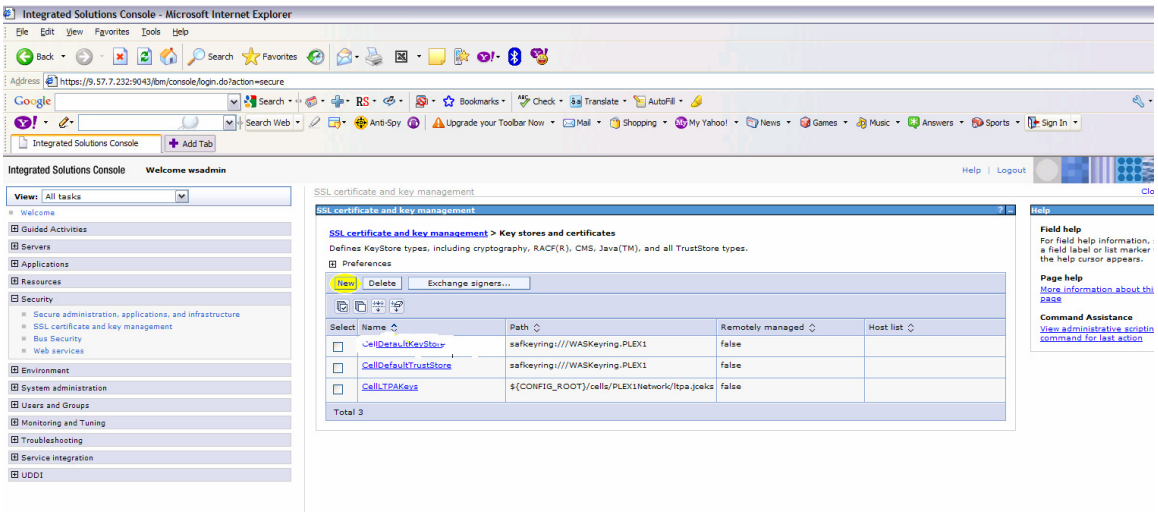
Configuring file based Keystores in WebSphere Application Server z/OS cell

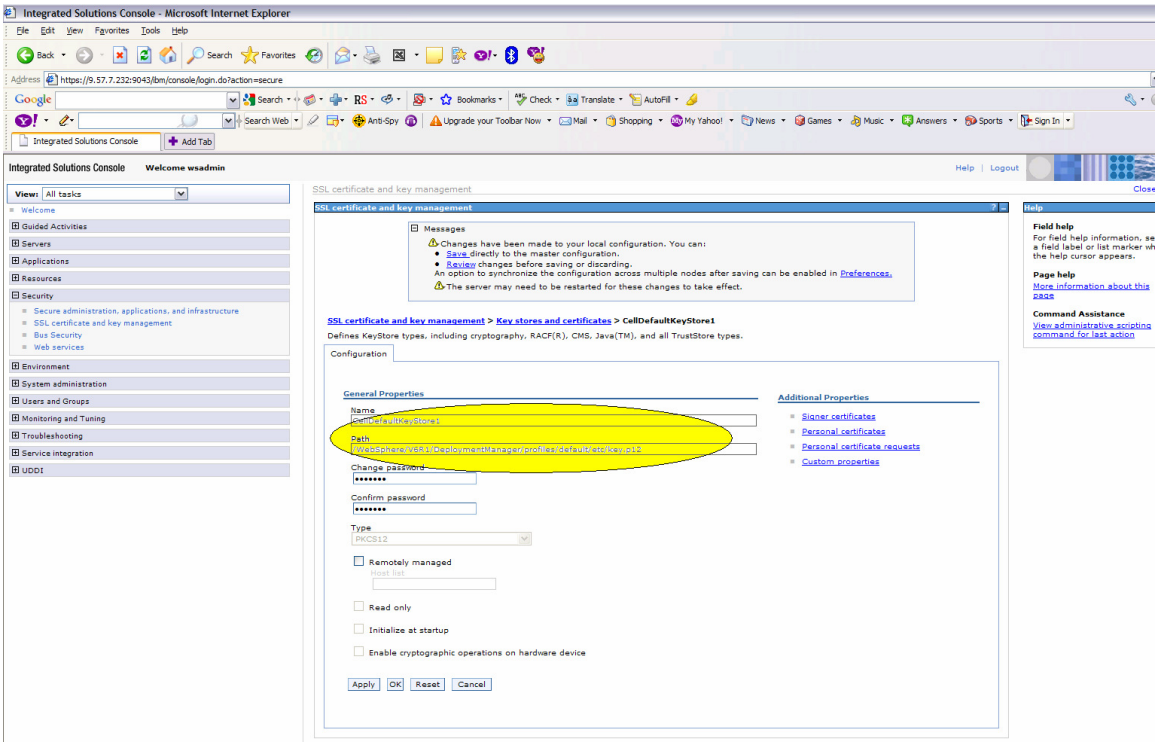
This procedure describes the steps to configure, setup to use file-based Keystores instead of SAF Keyring to a zWebSphere Application Server cell similar to the distributed WebSphere Application Server Environment.

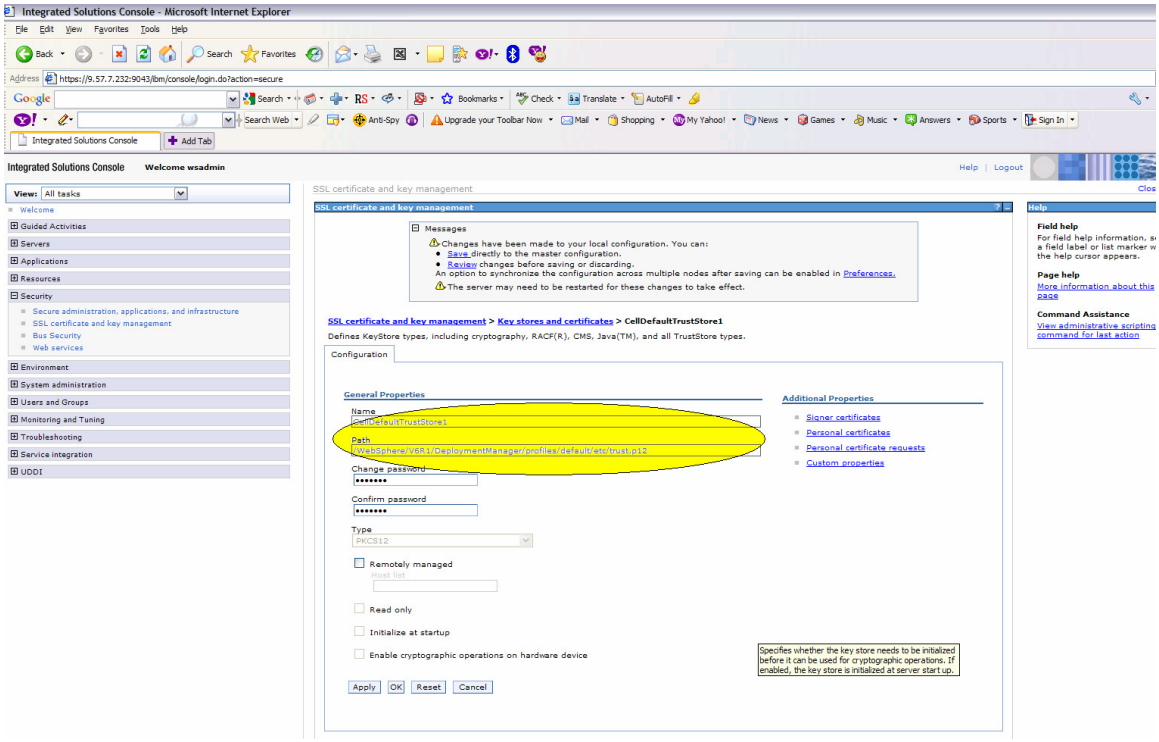
Below are the steps to change the **CellDefaultSSLSettings , NodeDefaultSSLSettings** to point to the file-based keystores instead of the SAF Keyring for the cell and node/s under the cell.

Create new cell keystores to point to the keystore files

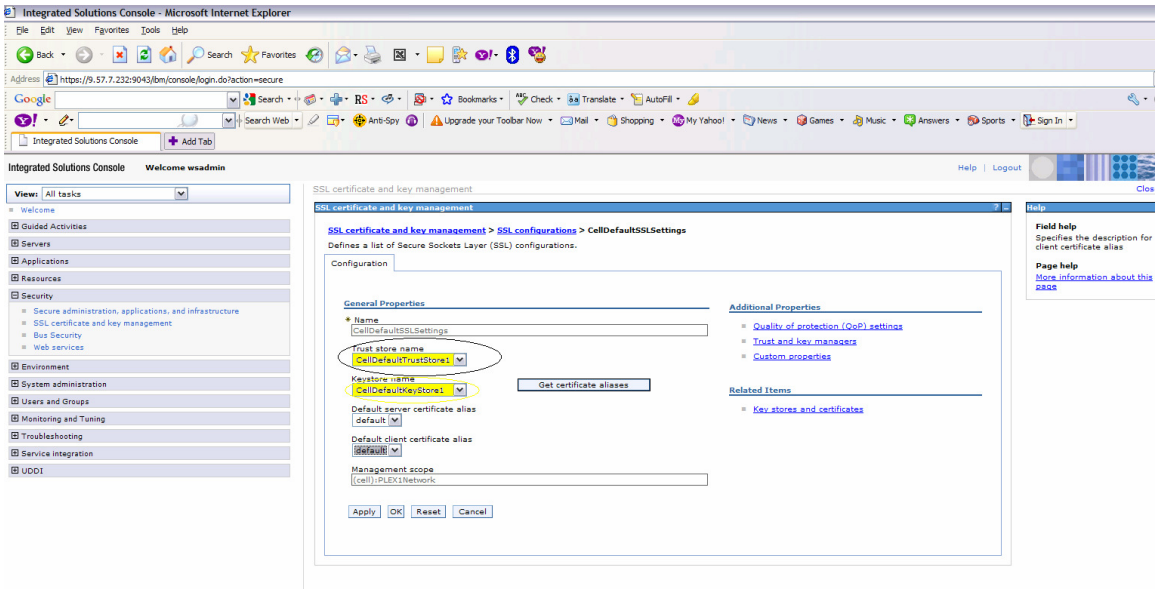
> **Note:** You can use the existing **CellDefaultKeyStore**, **CellDefaultTrustStore** to point to the keystore files instead of the SAF Keyring but we've decided to create a new one instead. The password for the keystore , truststore files is **WebAS.**

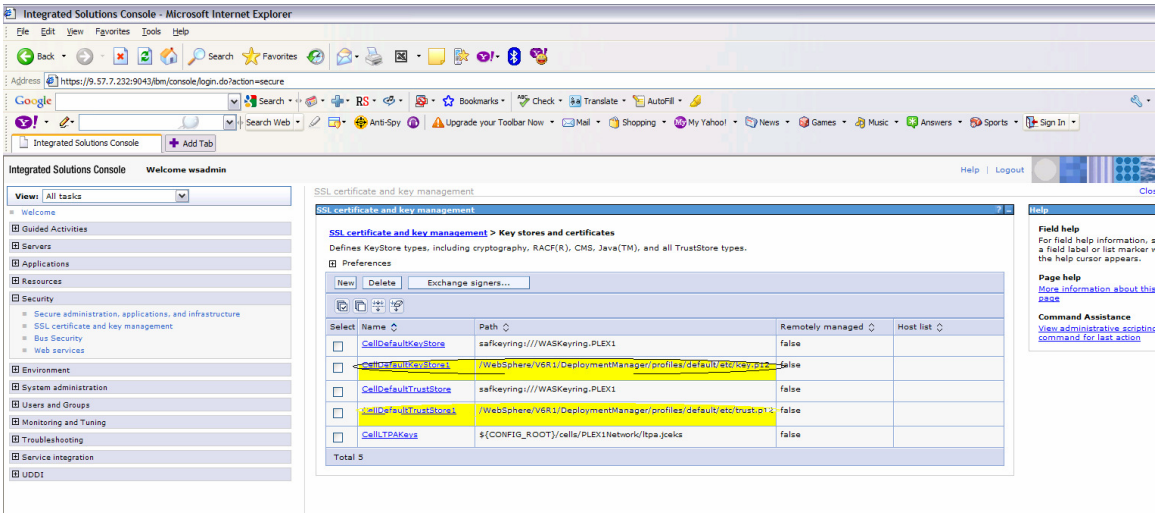

Integrated Solutions Console - Microsoft Internet Explorer
Integrated Solutions Console
Welcome wsadmin
SSL certificate and key management
SSL certificate and key management > Key stores and certificates
Defines KeyStore types, including cryptography, RACF(R), CMS, Java(TM), and all TrustStore types.
Preferences
New
Delete
Exchange signers...
Select
Name
Path
Remotely managed
Host list
CellDefaultKeyStore
safkeyring:///WASKeyring.PLEX1
false
CellDefaultTrustStore
safkeyring:///WASKeyring.PLEX1
false
CellLTPAKeys
${CONFIG_ROOT}/cells/PLEX1Network/ltpa.jceks
false
Total 3


Integrated Solutions Console - Microsoft Internet Explorer
Integrated Solutions Console
Welcome wsadmin
Messages
Changes have been made to your local configuration. You can:
Save directly to the master configuration.
Review changes before saving or discarding.
An option to synchronize the configuration across multiple nodes after saving can be enabled in Preferences.
The server may need to be restarted for these changes to take effect.
SSL certificate and key management > Key stores and certificates > CellDefaultKeyStore1
Defines KeyStore types, including cryptography, RACF(R), CMS, Java(TM), and all TrustStore types.
Configuration
General Properties
Name
CellDefaultKeyStore1
Path
${WebSphere:cell}/DeploymentManager/profiles/default/etc/key.p12
Change password
Confirm password
Type
PKCS12
Remotely managed
Host list
Read only
Initialize at startup
Enable cryptographic operations on hardware device
Additional Properties
Signer certificates
Personal certificates
Personal certificate requests
Custom properties
Apply
OK
Reset
Cancel



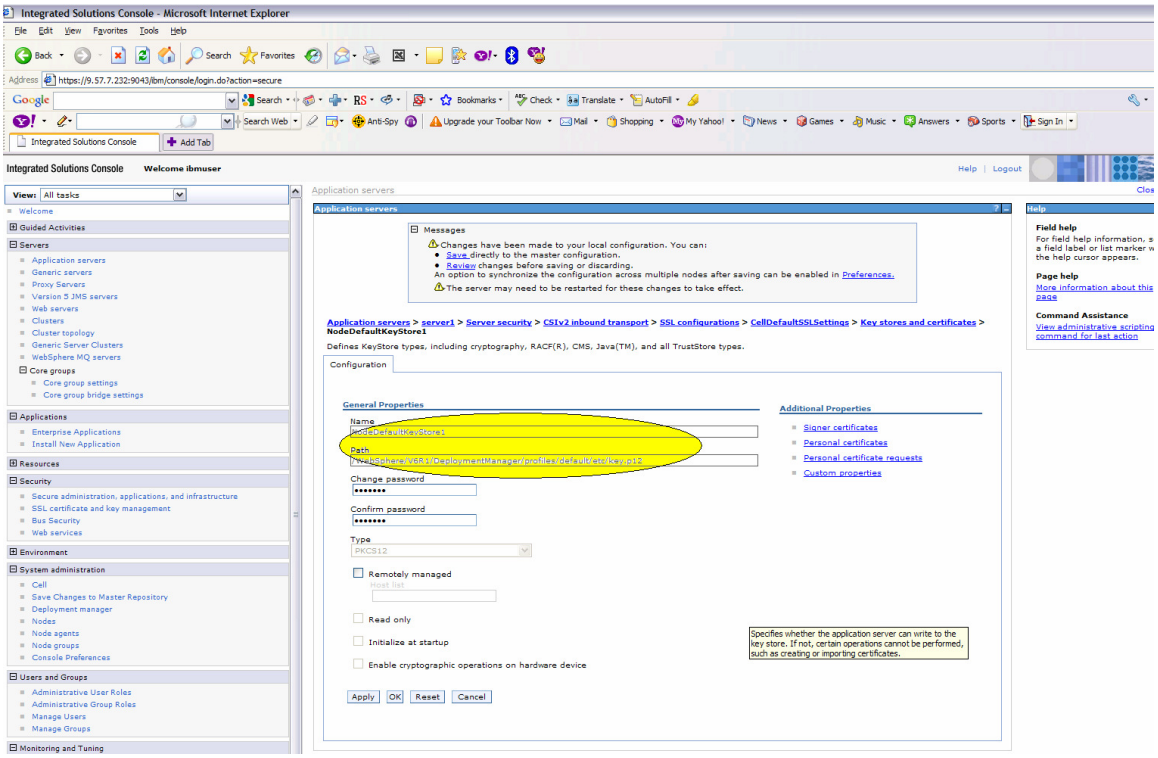
Configure the CellDefaultSSLSettings to the new keystores

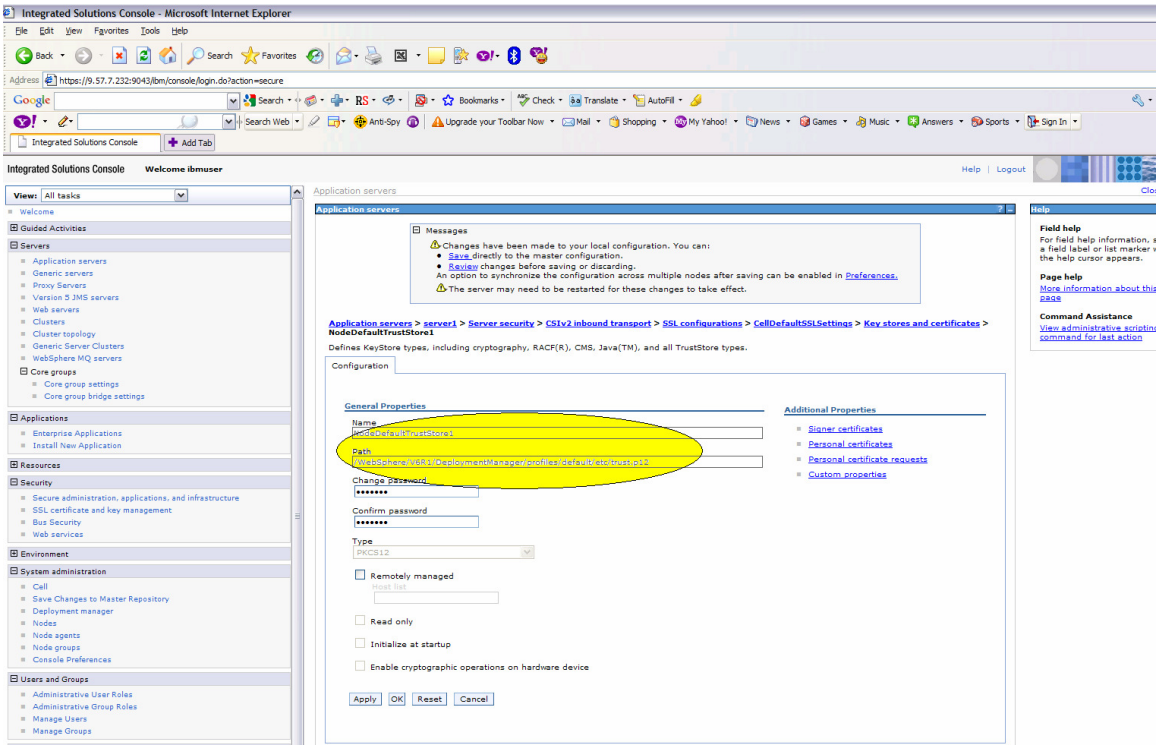


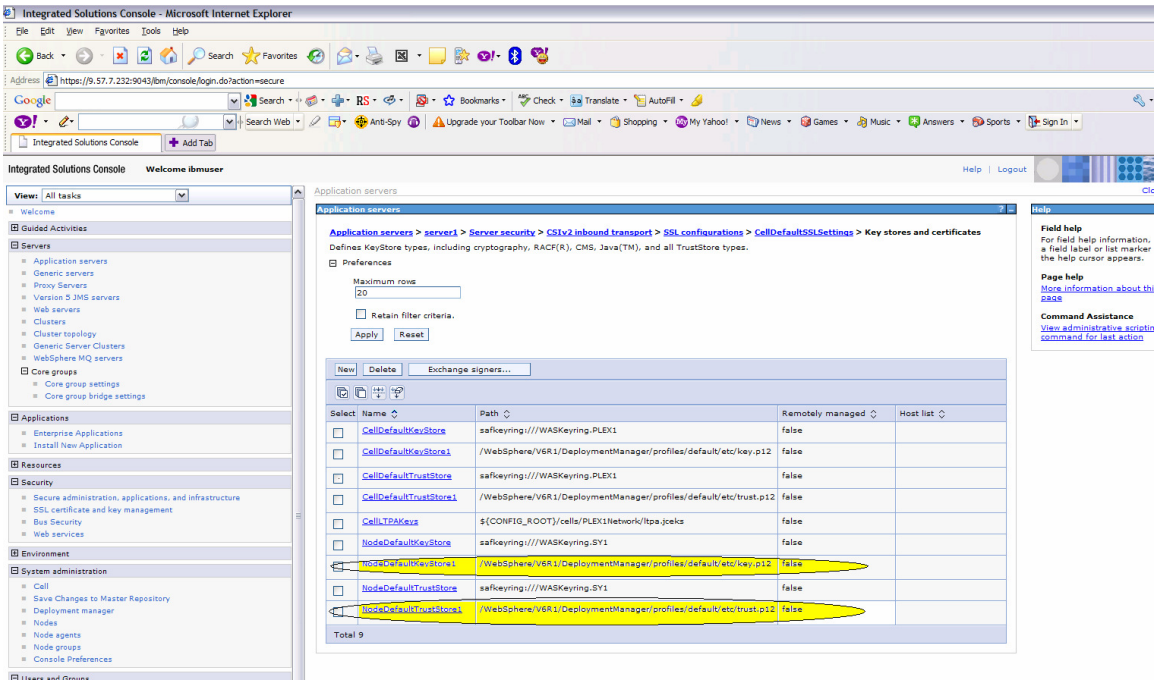


Create new node keystores to point to the keystore files

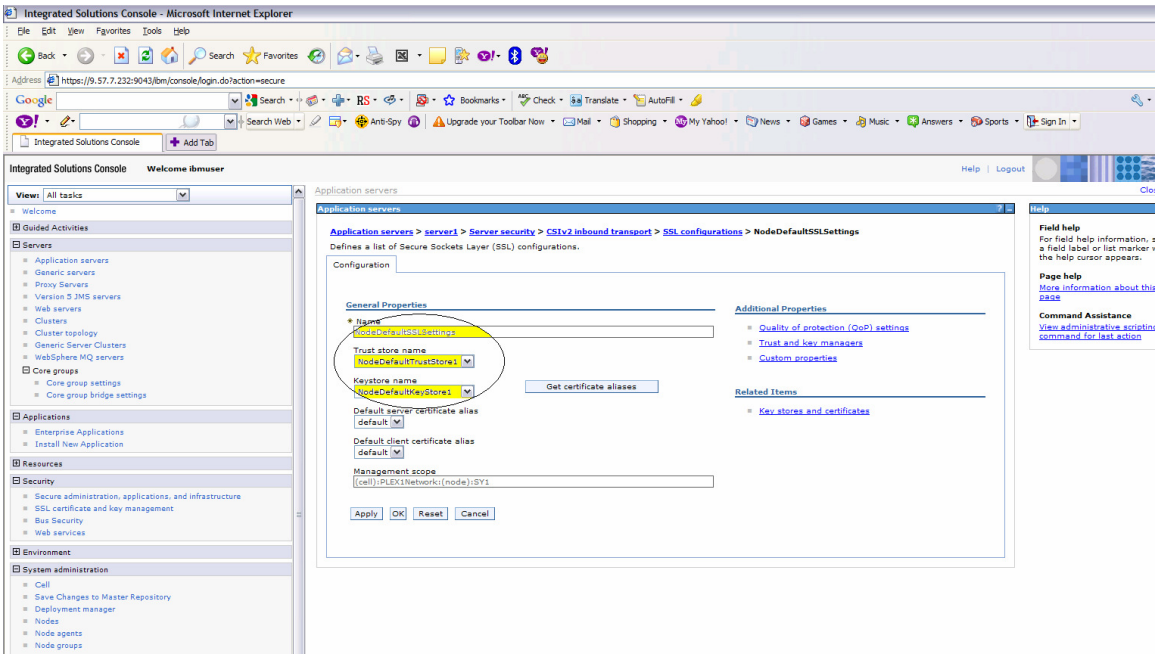
Note: You can use the existing **NodeDefaultKeyStore**, **NodeDefaultTrustStore** to point to the keystore files instead of the SAF Keyring but we've decided to create a new one instead. The password for the keystore , truststore files is **WebAS**







Configure the NodeDefaultSSLSettings to the new keystores



Save changes and sync the Master Configuration and upon restart, the z/OS WebSphere Application Server cell will be using the file based keystores .

Document Change History

Check the date in the footer of the document for the version of the document.

November 06, 2009	Original document.
November 08, 2009	Republished with assigned Techdoc number.

End of WP101579