WebSphere Application Server
for z/OS and OS/390

Preparing WebSphere Application Server for z/OS for Global Security

Bob Teichman - TEICHMN@US.IBM.COM
IBM Americas Advanced Technical Support -- Washington Systems Center
Gaithersburg, MD, USA

Session W09

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

AIX*
CICS*
e-business logo*
IBM*
IBM eServer
IBM logo*
IMS
OS/390*
RACF*
S/390*
WebSphere*
z/OS*
zSeries*
* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

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.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

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

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

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- **WebSphere Global Security**
- **SSL, WebSphere and RACF**
- **Comparing RACF, CA-Top Secret and CA-ACF2 Keyrings and Certificates**
- **Troubleshooting tips**

This presentation is divided into several sections:

- A quick overview of what Global Security is
- Information on how SSL works,
- A comparison of the command structure and restrictions in RACF versus CA-ACF2 and CA-Topsecret
- AT the end of this presentation I'll cover some troubleshooting tips as well as how to turn security off in the event that you can not even get into the Admin console.

Version Date: September 2, 2005

What's Global Security?

- **One big WebSphere 'switch' that activates many settings related to WebSphere security. The settings include:**
  - User Registry
  - Authentication Mechanisms
  - Secure Sockets Layer (SSL)
  - Administrative Console Security
  - Other Misc. Security Stuff
- **Global Security is specified at the administrative console, and indicated in the security.xml file.**

Global Security is a setting, turned on or off via the Admin Console, that tells the cell to operate according to the values set in the security options.

Most of the security options for WebSphere servers, as well as applications, are stored in xml files.

Any security options that are set are ignored until Global Security is turned on.

Some of the settings include:

- SSL keyring names
- What the user registry is (for validating userids and passwords), authentication, application security.
- Whether SAF is used to enforce EJBROLES
- Etc..

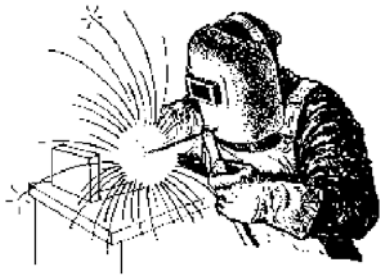
Global security is turned on through the admin console, and turned off either through the Admin console or the administrative scripts, wsadmin.

With Global Security turned off, which is the initial setting, anyone having network access to your system can access the Admin Console and modify the WebSphere configuration.

Turning Global Security off requires WebSphere administrator authority.

When to Activate Global Security

- After WebSphere is up and running, and you are familiar with WebSphere and the Administrative Console, it's time to activate Global Security.
- Global Security must be activated in order to secure the Administrative Console.
- Global Security must be activated in order to secure any applications (Basic Authentication, Form Based, etc.).
- Don't wait until it's time to go into production to enable Global Security!



It is advisable to run your WebSphere cell without security for a while so that you can familiarize yourself with the environment and the console before you start testing security.

In order to restrict access to the Admin Console Global Security must be enabled.

Any security specifications in your applications, such as:

- Authentication methods
- Authorization based on EJB roles

Will not be enabled until Global Security is turned on.

Potential Problem

- **An opportunity for recognition.**
  - Remember it is possible
    - to lock everyone out of WebSphere
    - not be able to start your servers and
    - even not be able to even login to the administrative console or to turn Global Security off.
  - Be prepared for the worst case scenario.
    - Back up your HFS

Turning global security on forces SSL communication between the WebSphere address spaces and enables admin console security.

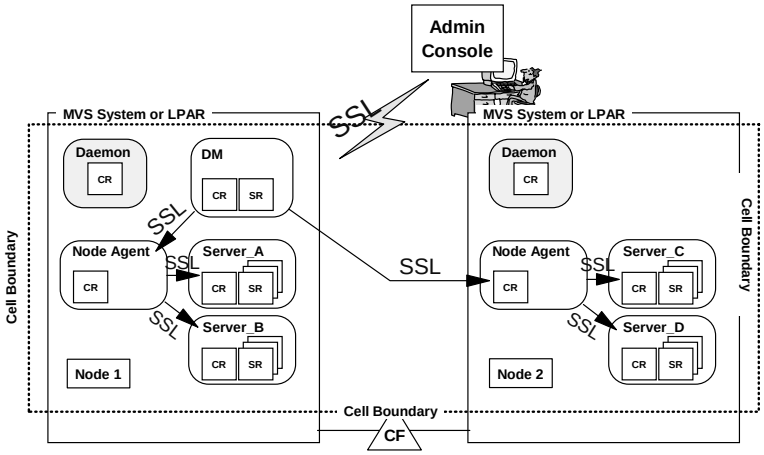
If some part of the SSL setup is incorrect, WebSphere might not be able to communicate across its' address spaces and the entire runtime might not come up successfully.

Unfortunately, it's also possible to lock yourself out of the system. In other words, you can not get to the Admin Console to turn security off.

So we're going to discuss how to enable global security without shooting yourself in the foot.

And how to disable it if you do.

Global Security and SSL



- **Enabling Global Security activates SSL for most WebSphere communication:**
  - Use of the Admin Console.
    - Requests to the HTTP port will be redirected to the SSL port.
    - Access to the console requires the administrator's userid and password.
  - WebSphere Inter-server Communication flows
    - Admin flows between servers.
    - Distributed application calls between servers.

The most noticeable effect of enabling global security is that it turns on SSL for so many things.

With Global Security on, http requests to the admin console application will be automatically redirected to the SSL port. This is to protect the admin userid and password, which you will now be prompted for.

WebSphere will do the same redirect to an SSL port for users of your business applications, if you deploy the applications specifying that SSL is required.

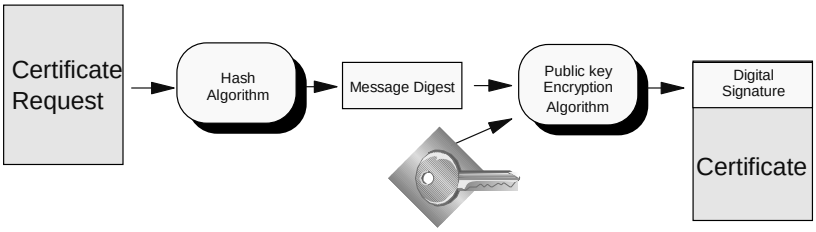
The administrative (SOAP/JMX) flows between the Deployment Manager, the Node Agents and the Application Servers are also SSL encrypted.

And calls between application servers (CSIv2 and zSAS) are encrypted as well.

Digital Certificates

SSL implies the use of

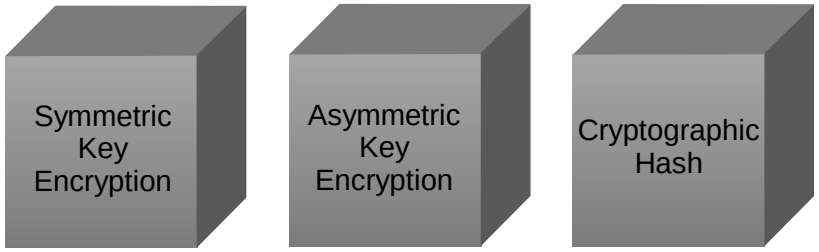
- Digital certificates
  - signed by a trusted third party
- Cryptography
  - to encrypt the information transmitted
  - to authenticate the parties and the certificates that are used



Use of SSL implies that servers must present digital certificates to clients. These certificates must be signed by a trusted third party, known as a Certification Authority.

SSL uses cryptography technologies to protect the data and authenticate the Certificates.

The Three Building Blocks of Cryptography



Symmetric-Key Encryption

Both parties use the same key

So key distribution requires a separate (secure) messenger

Asymmetric-Key Encryption

Also Known as Public Key cryptography

Two keys are used, the public key and private key

Only the public key can decrypt messages encrypted with the private key.

Only the private key can decrypt messages encrypted with the public key.

Cryptographic Hash

A one-way function where data is reduced to small numeric value
a message digest

Cryptography is used in several ways in SSL communications.

Asymmetric key encryption

- Used in the initial handshake between client and server
- Two keys are used, the public key and private key
- Only the public key can decrypt messages encrypted with the private key.
- Only the private key can decrypt messages encrypted with the public key.

Symmetric key encryption

- Each party (the sender and the receiver) must have the same key.
  - This raises the issue of how to distribute keys securely.
- The key must only be shared by the parties involved.
- The key must be long enough.
  - An encrypted message can be decrypted eventually if every possible key is tried. Known as a brute force attack.
  - This explains the move from DES to Triple DES.
- The keys must be changed occasionally to stay ahead of a brute force attack.
- Used by the client and server in further communications.

Cryptographic Hash

- Used by the CA to create the digital signature and used by the client to validate the signature.
- A one-way function where any amount of data is reduced to small numeric value called a message digest.
- It is computationally unfeasible to construct a message that yields a given message digest.
- It is extremely unlikely that two different messages will yield the same message digest.
- Changing one bit of the message results in, on average, a change in half the bits in the message digest.

The Certification Authority (CA)

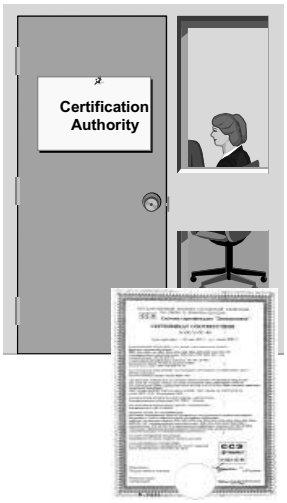
The CA is an entity trusted by both the client and the server.

The job of the CA is to authenticate subjects (servers, clients) and issue identity credentials (certificates).

The CA issues a credential (a certificate) to the server, that associates the server's name with the server's public key.

The client and server trust that the CA will not issue a certificate to an imposter.

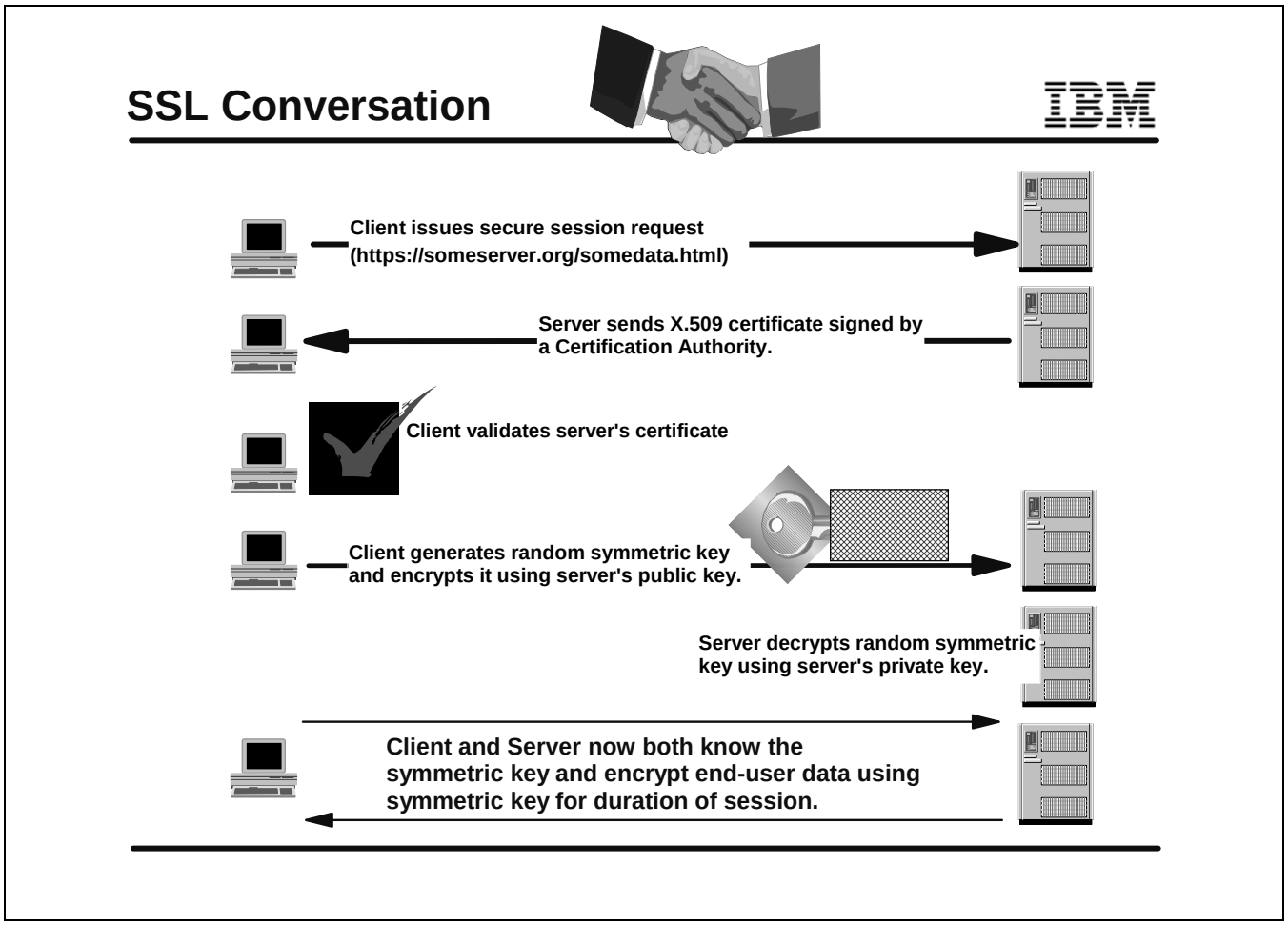
The client can validate the server's certificate at any time. Validating a certificate proves that it's authentic and has not been modified.



The fundamental issue that we are trying to solve is the issue of trust. Can I, the client, trust that I am talking to a legitimate server.

In order for this to occur, in SSL communication we use a trusted third party, the certification authority, to certify that, in this case, the provider of a service, the server, is who they say they are.

The CA is universally trusted to verify the identity of the server and provide that server with a set of credentials that the client can be assured is authentic.



This diagram shows the highlights of what occurs in a conversation between a client and a server using SSL.

- The client makes a request over the SSL port
- The server presents its' credentials in the form of a digital certificate. This certificate is signed by some 'Certification Authority' who is basically vouching for the servers' authenticity.
- The client validates the certificate by using the 'Certification Authority's' public key to decrypt information in the digital signature.
- The client gets the CA's public key from the CA's certificate stored on its' (the clients') keyring. (more on this later)
- The client generates a random key and encrypts it with the servers' public key and sends it to the server.
- The server decrypts the key sent by the client using its' private key.
- This symmetric key will now be used by both the server and the client to encrypt any further communications between them.

When the server's cert is not valid...

- If the CA cert is not in the client's browser, or the browser can't validate the server's cert, the client gets this popup warning. Clicking Yes to continue is a dangerous habit.
  - This is why customers use commercial CAs like Verisign, which is already in IE and other browsers.

When the server presents its' certificate to the client, the client will retrieve the CA's certificate with its' public key from its' keyring. It will use the public key to decrypt the cryptographic hash in the signature to make sure that it matches what is contained in the certificate on its' keyring. If the hash does not match or the client does not have the CA's certificate on its' keyring, the CA is deemed to be not trusted.

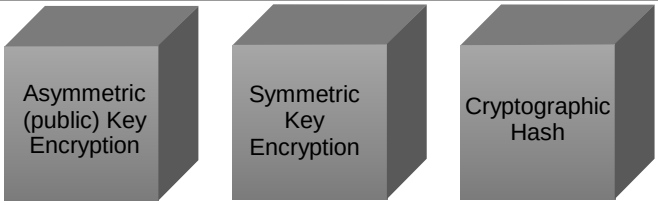
All browsers are shipped with keyrings containing the certificates of all the well known accepted Certification Authorities.

Browsers will present a pop-up warning to the client indicating that the certificate is issued (signed) by a company that you have chosen not to trust.

The client can choose to accept it or not (not a good idea).

Therefore, it is important that you obtain certificates for your servers from generally recognized and accepted Certification Authorities.

SSL makes good use of cryptography



The Three Building Blocks of Cryptography

- **Asymetric Public key cryptography**
  - **Used by the Certification Authority(CA) to 'sign' the server's cert.**
  - **Used by the client to validate the server's cert.**
  - **Used by the client to communicate a symmetric key value to the server.**
- **Symmetric key cryptography**
  - **Used by the client and server for ongoing conversation.**
- **Cryptographic hash**
  - **Used by the CA to create the digital signature on the server's cert.**
  - **Used by the client to validate the server's cert.**

Cryptography is used in several ways in SSL communications.

Asymmetric key encryption is used in the initial handshake between client and server

- The CA uses a **cryptographic hash** to create the Digital Signature and then encrypts the Digital Signature using its' private key.
- The client uses the public key to decrypt the signature.
- The client then uses the same cryptographic hash in validating the signature.

Once the handshake is successfully completed, a **symmetric key** is created by the client and is then used in all further communication.

In order to securely communicate this key to the server, the client encrypts the key with the servers public key. This can only be decrypted by the holder of the private key, which is only the server.

SSL Keyrings

- **Certificates are connected to Keyrings.**
- **Keyrings are stored in RACF**
  - Identified by UserID (owner)
    - unique by UserID
  - Has a name (label)
- **A Server Keyring Should contain**
  - The Server's Certificate, signed by some Certificate Authority
    - The client gets the server's public key from the server's cert.
  - The Server's Private Key (you can't display this).
  - The Certificate of the CA that signed the Server's Cert.
- **A Client Keyring Should contain**
  - Certificate of the CA that signed the Server's Cert.
    - If a CA Cert isn't in the client's keyring, that CA is **not trusted** by the client.
    - The client gets the CA's public key from the CA cert.
  - If SSL Client authentication is specified, the client needs a certificate and private key too.
    - CA's can issue certificates to clients, too.

SSL Certificates are stored in the RACF data base in a construct called a keyring.

- Keyrings are owned by a userid and has a name associated with it.
- A user may have many keyrings, each with a different name.
- Many users can have keyrings with the same name.

Keyrings for servers must contain:

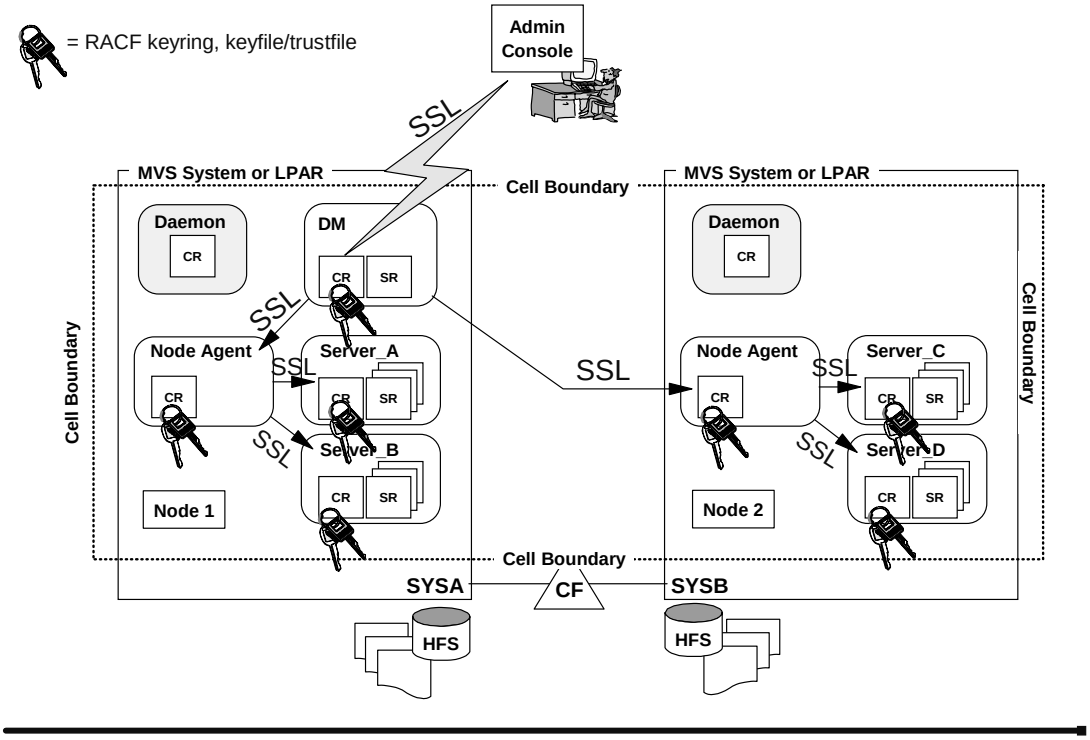
- Private certificate signed by a certification authority. This is the certificate that is sent to the client when an SSL request is made.
- The certificate of the Certification Authority that signed the servers' certificate.

Keyrings for clients

- Clients, such as your browser or the WebSphere administrator, also require keyrings.
- Their keyrings must contain the certificates of the Certification Authorities that it trusts.
- These are used to obtain the CA's public key so that they can validate the digital signatures on the server certificates they are presented with.

If you are using SSL client authentication, then the clients' keyring also needs to contain its' private certificate that it will present to the server for authentication.

SSL in an ND Configuration



As mentioned earlier, once Global Security is enabled, all communications between the administrator client and WebSphere must be over SSL. In addition, all communications between the Deployment Manager and the Node Agents and between the Node Agents and the Application Servers is also over SSL.

In order to successfully communicate, there must be certificates and keyrings for the userids involved.

SSL Keyrings and WebSphere

- In WebSphere V5, only Controller regions needed Keyrings.
  - Daemon, DM Controller, NA, App Server Controller.
  - These regions need a personal cert and a CA cert.
  - Used for HTTPS traffic as well as JMX over SOAP w/ SSL.
- In WebSphere V6, Servant regions also need a keyring.
  - Servants now use JMX over SOAP w/SSL.
  - The servants need a keyring with the CA cert in it. (A 'client' keyring)
  - Created by the BBOCBRAK and BBODBRAK jobs.
- In WebSphere V6, the Control Region Adjunct regions also need a keyring.
  - CAR is a special purpose servant used for messaging services.
  - The CAR needs a keyring with the CA cert in it. (A 'client' keyring)
  - Created by the BBOCBRAK and BBODBRAK jobs, unless you migrate.

WebSphere V5

- Deployment Manager, Node Agent and Application Server Controller are all servers in the SSL conversation.
- The userids that these address spaces run under must have a servers keyring.
  - Private certificate signed by a CA
  - The CA's certificate
- Servant regions do not require a keyring
- Administrator userid requires a client keyring
- The certificate of the CA that signed the servers certificate.

WebSphere V6

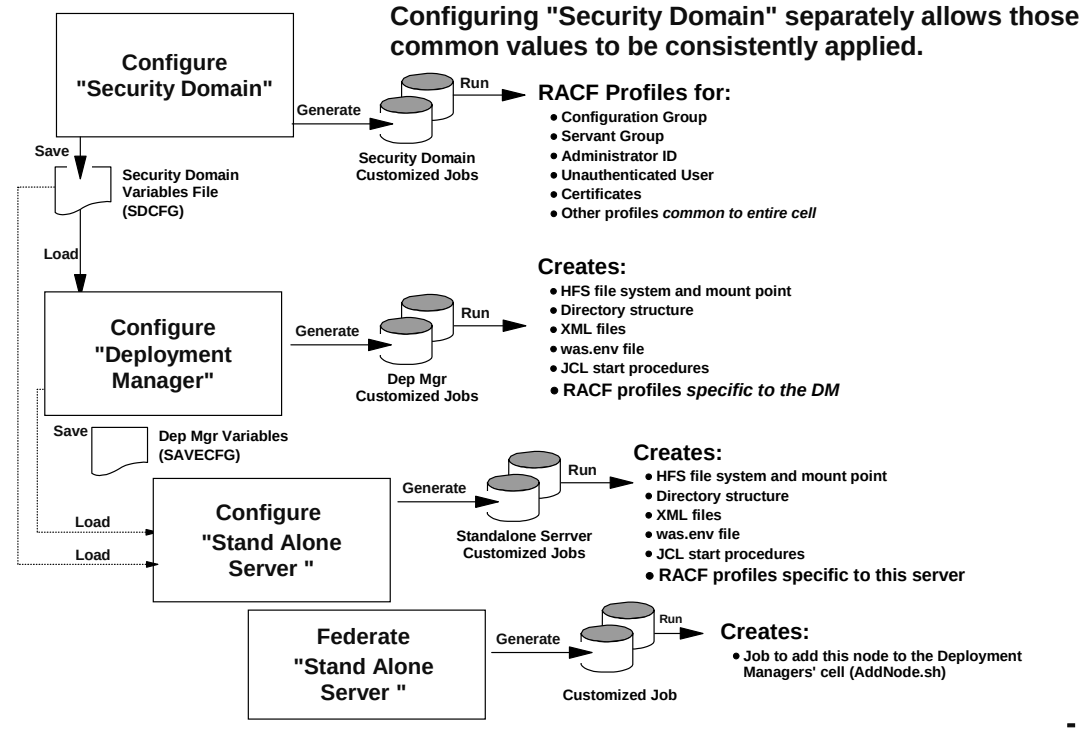
- The same requirements as V5 plus...
- The userid that the servant address space runs under needs a client keyring.
  - The certificate of the CA that signed the servers certificate
- The userid that the adjunct address space runs under needs a client keyring.
  - The certificate of the CA that signed the servers certificate

ISPF Dialogs

- Create keyrings for the various userids
- Optionally create certificates and place them on the keyrings

We'll discuss when and how these are created in the next few charts.

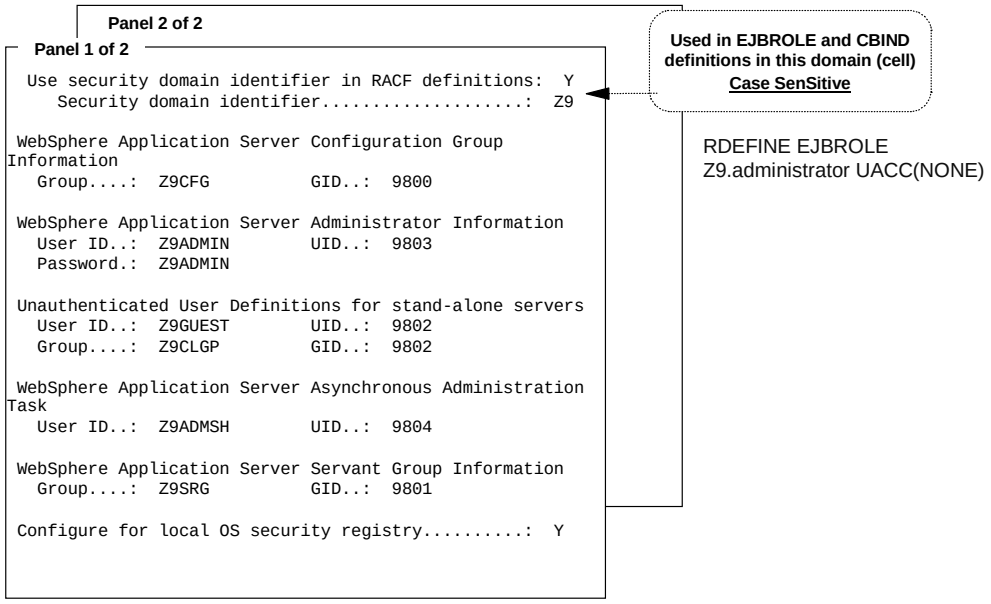
Configuring A WebSphere Cell



In the creation of a WebSphere cell, there are several jobs that get run that define various RACF profiles.

The dialogs will also generate the RACF commands to define certificates and keyrings depending on some options you specify in the dialogs.

Creating the Security Domain

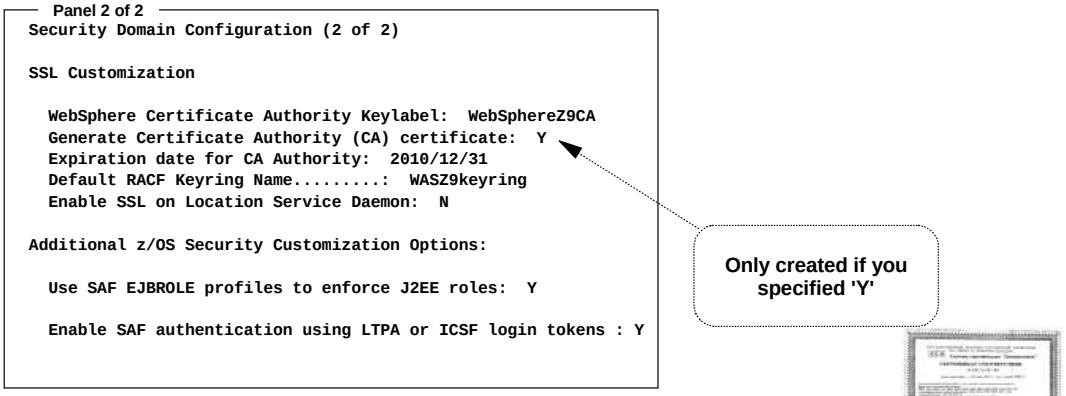


Two panels that capture information about the security definitions common to the planned cell, not just the Base AppServer.

When you define the Security Domain, you specify a userid for the WebSphere administrator.

The keyring for the administrator ID is not created at this point in time, but is created when you run the RCAF jobs for the Deployment Manager.

Certification Authority (CA) Certificate



```
/* Create CA Certificate for WebSphere Security Domain

RACDCERT CERTAUTH GENCERT -
 SUBJECTSDN(CN('WAS CertAuth for Security Domain')  OU('Z9Cell.WebSphere for zOS')) -
 WITHLABEL('WebSphereZ9CA')  -
 TRUST NOTAFTER(DATE(2010/12/31))
```

On the second panel of the Security Domain definitions, you have an option to generate a Certification Authority certificate. This certificate can and will be used to sign the certificates for the controller address spaces and will be connected to the all of the keyrings.

This makes your RACF system the certification authority. Shown here is the RACF command generated to define the CA certificate.

The problem with that is that clients such as:

- Browsers
- Other Java applications in other servers or systems

Would not recognize your RACF system as a valid CA and in the case of the browsers would pop up a warning, and in the case of other programs would fail the request.

The administrator, servant and adjunct would recognize the CA, since the dialogs would generate the commands to add this certificate to their keyring.

In a testing environment or in an intranet application where the client community is tightly controlled this approach might work.

In any production environment, with a large number of users whether they be inside your enterprise or outside, you would instead purchase a certificate from a recognized Certification Authority. In that case you would specify ‘N’ in the dialogs for this option.

The keyring name specified on this panel is case sensitive. This name will be used for all keyrings in the cell. All keyrings in a cell must use the same keyring name.

Access to Keyrings

- **FACILITY class profiles are generated to permit servers access to their keyrings and certificates. It's not enough for the server to own the certificate and keyring!**
  - Access by a server to its cert:
    RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)

    PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(Z9CFG) - ACC(READ)
  - Access by a server to its keyring:
    RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)

    PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) - ID(Z9CFG) ACC(READ)

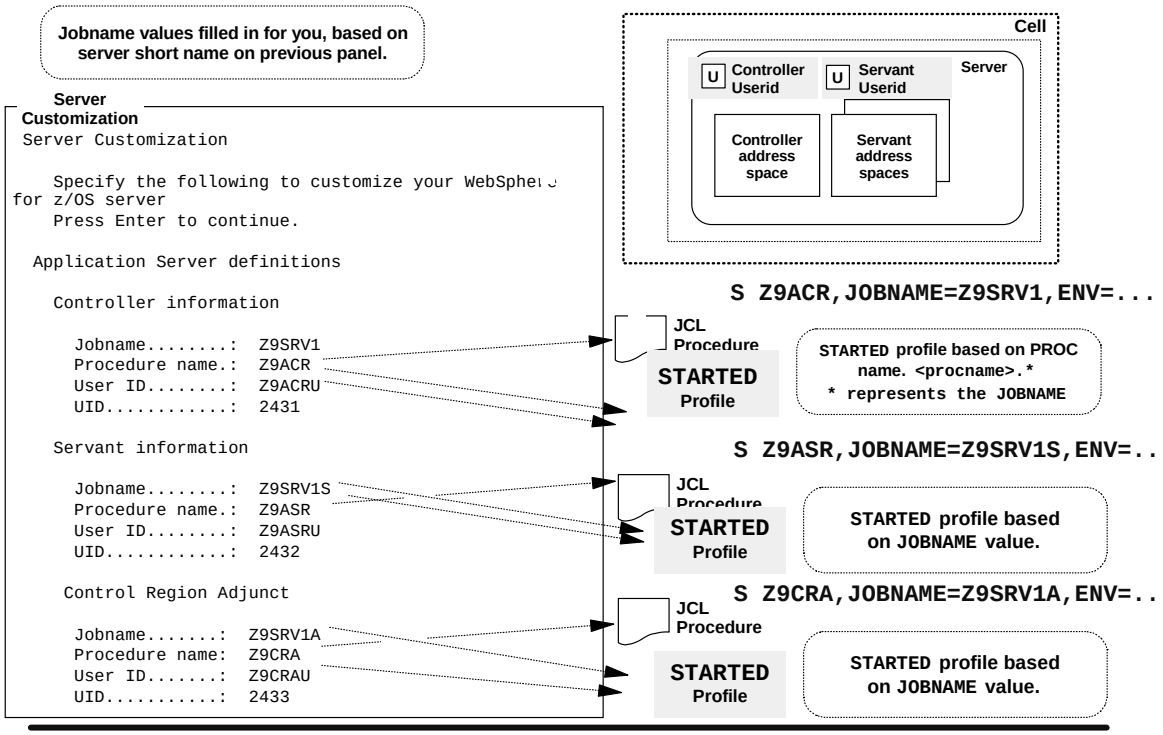
Access to keyrings and certificates is protected by RACF by a set of profiles in the 'FACILITY' class.

Even though the keyring is associated with the users' ID, the user must have 'READ' authority to the IRR.DIGTCERT.LISTRING profile in order to access its' keyring.

The user must also have 'READ' access to the IRR.DIGTCERT.LIST profile to be able to access its' certificate.

Notice that permission to these profiles is specified to the configuration group ID defined for your cell. This gives permission to all of the address spaces in the cell as well as the cell administrator.

Server Customization, Panel 2 of 6



The information provided on the server customization panel, whether it's the Standalone Server, the Deployment Manager or when defining the Empty Managed Node, is used to create user profiles and 'STARTED' class profiles for the associated address spaces.

In addition, these userids are used to generate the commands to add keyrings and certificates for these users.

Keyring and Certificate

- **The BBOWBRAK job creates the keyring and server certificate used by the app server signed by the cell CA.**
  - **RACDCERT ADDRING(WASZ9Keyring) ID(Z9ACRU)**
  - **RACDCERT ID (Z9ACRU) GENCERT - SUBJECTSDN(CN('Z9ACRU.Z9SR01') O('IBM') OU('Z9')) - WITHLABEL('DefaultWASCert.Z9SR01') - SIGNWITH(CERTAUTH LABEL('WebSphereCAZ9')) - NOTAFTER(DATE(2010/12/31))**
  - **RACDCERT ID(Z9ACRU) CONNECT - (LABEL('DefaultWASCert.Z9SR01') RING(WASZ9Keyring)- DEFAULT)**
  - **RACDCERT ID(Z9ACRU) CONNECT (RING(WASZ9Keyring)- LABEL ('WebSphereCAZ9') CERTAUTH)**

- The first command above define a keyring for the controllers' userid, Z9ACRU.
- The next command defines a certificate for the controller userid and signs it with the CA certificate that was created when the Security Domain was defined. This certificate will only be created if you specified 'Y' on the Security Domain panel in answer to the question:
  - Create a Certification Authority Certificate.
- These certificates are then added to the keyring.
- The certificates are referenced by the 'LABEL', name, under which they were created.

It is important to note that keyring names are case sensitive and can be as long as 237 characters.

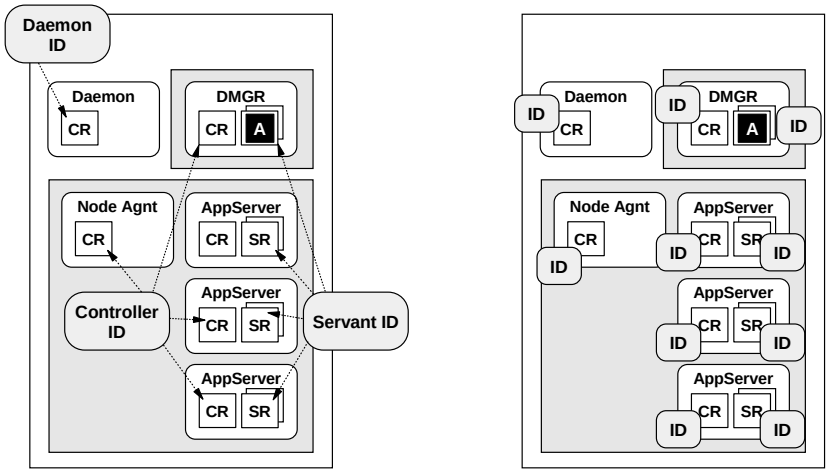
Administrators' Keyring

- A keyring is also created for the Admin userids
  - **RACDCERT ADDRING(WASZ9Keyring) ID(Z9ADMIN)**
  - **RACDCERT ADDRING(WASZ9Keyring) ID(Z9ADMSH)**
  - **RACDCERT ID(Z9ADMIN) CONNECT (RING(WASZ9Keyring) - LABEL ('WebSphereCAZ9') CERTAUTH)**
  - **RACDCERT ID(Z9ADMSH) CONNECT (RING(WASZ9Keyring) - LABEL ('WebSphereCAZ9') CERTAUTH)**

The dialogs generate commands to create keyrings for the two administrator IDs and places the CA certificate on their keyrings.

User ID Strategies

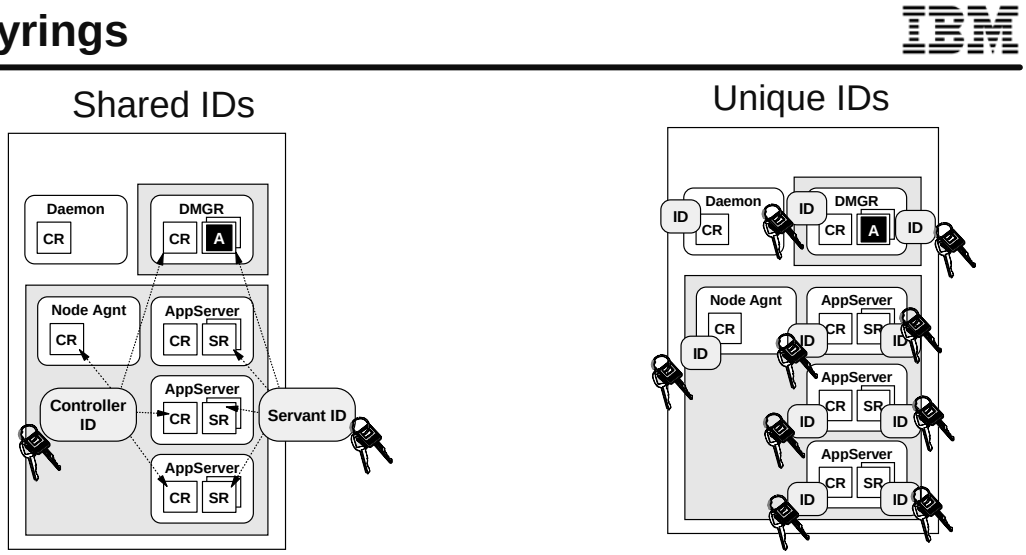
There two approaches one can take when considering the user IDs used when running the servers: a small set used by all, or separate for each:



There are two different approaches to assigning userids to WebSphere address spaces. The assumed default approach is to run each of the address spaces under a unique userid. This is what the ISPF dialogs are designed to do.

The other approach, which is becoming more common, is to run all of the controller address spaces, under one ID, and all of the servant and adjunct address spaces under a second ID. This approach doesn't provide as much granularity in your security environment, but does simplify things some when it comes to keyrings and certificates.

Keyrings



Use the same RACF userid for all the controller regions.
- One keyring and cert will be shared by all servers.

Use a second RACF userid for the Servant regions.
- The Servant regions will share a 'client' keyring.

Each address space will need its' own keyring.

In the diagram on the right, each controller address space would have its' own keyring defined, and have the appropriate certificates connected to those keyrings. This would result in a total of 9 keyrings for this configuration.

In the diagram on the right, since the address space are running under common IDs the keyrings could be shared. This would mean that you would need to define only two keyrings. One a server keyring shared by the controller address spaces and one client keyring shared by the servant address spaces.

Not shown on this chart are the 'adjunct' address spaces used for JMS. They could share the keyring used by the servant address space in either of the configurations.

If you are going to use the approach on the left of shared userids and keyrings, then be aware that the ISPF dialogs will generate many duplicate commands, some of which will fail. It would be advisable that after you generate the commands using the BBOxRAJ jobs, you would then edit the member in the .DATA dataset which is input to the BBOxRAK jobs.

RACF, CA-Top Secret and CA-ACF2

A comparison of Keyring and Certificate commands for RACF, CA-Top Secret and CA-ACF2.

If you are not using RACF in your environment, but using one of the other SAF security products, you will have to hand code the commands for your environment.

This next section of the presentation is an attempt to show the comparable commands for CA-ACF2 and CA-Topsecret used for the creation of keyrings and certificates.

The actual commands shown have not been run in the respective environments. Please check the syntax carefully before using these in your environment.

CA-Top Secret Keyrings

- **A command to create a RACF Keyring:**
  - RACDCERT ID(MKACRU) ADDRING(MKKeyring)
    - This creates a keyring named MKKeyring, owned by user MKACRU.
    - The keyring name is case sensitive and can be up to 237 characters.
    - Different users can have a keyring with the same name, but a user cannot have multiple keyrings with the same name.
- **An equivalent CA-Top Secret Keyring:**
  - TSS ADD(MKACRU) KEYRING(MKACRU1) - LABLRING('MKKeyring')
    - This creates a keyring named MKKeyring, owned by user MKACRU.
    - The KEYRING name is limited to 8 chars, and must be specified.
    - The LABLRING is an alternate name, case sensitive and can be up to 237 characters. WebSphere calls the keyring by its LABLRING name.
    - Like RACF, different users can have a keyring with the same name, but a user cannot have multiple keyrings with the same name.

A very important point to note here is that CA-Topsecret has 2 names that are used for the keyring. The keyring name is limited to 8 characters in CA-Topsecret, but it allows for another name reference called ‘LABLRING’ which can be mixed case and up to 237 characters. This is the name that WebSphere uses to access its’ keyrings.

CA-ACF2 Keyrings

- **RACF create keyring**
  - RACDCERT ID(MKACRU) ADDRING(MKKeyring)
- **An equivalent CA-ACF2 Keyring:**
  - SET PROFILE(USER) DIV(KEYRING)
  - INSERT MKACRU RINGNAME(MKKeyring)
    - This creates a keyring named MKKeyring, owned by user MKACRU.
    - The RINGNAME is limited to 32 chars.
    - Like RACF, different users can have a keyring with the same name, but a user cannot have multiple keyrings with the same name.

The ring name in CA-ACF2 is limited to 32 characters.

Userids may only be associated with one keyring.

A RACF CA Certificate

- **A RACF command from BBOSBRAK to create a CA Certificate:**
  - RACDCERT CERTAUTH GENCERT - SUBJECTSDN(CN('WAS CertAuth for Security - Domain') OU('WebSphere for zOS')) - WITHLABEL('WebSphereCA') TRUST - NOTAFTER(DATE(2010/12/31))
    - Creates a CERTAUTH cert named WebSphereCA.
    - The cert name is case sensitive and can be up to 237 characters.
    - This cert is self-signed. It's a root CA certificate.
    - It's valid until the end of 2010.
  - Now for the equivalent Top Secret command...

This is the command that would be generated if your specified in the ISPF dialogs that you wanted to create a CA certificate that you will use to sign the server certificates.

CA-Top Secret CA Certificates

RACF CA certificate

```
   RACDCERT CERTAUTH GENCERT -
SUBJECTSDN(CN('WAS CertAuth for Security - Domain') -
OU('WebSphere for zOS'))  WITHLABEL('WebSphereCA')  TRUST -
NOTAFTER(DATE(2010/12/31))
```

- **The equivalent CA-Top Secret CA Certificate:**
  - TSS GENCERT(CERTAUTH) DIGICERT(CA001) - SUBJECTN('CN='"WAS CertAuth for Security Domain" - OU="WebSphere for z/OS"') NADATE(12/31/10) - KEYUSAGE('CERTSIGN') LABLCERT('WebSphereCA')
    - This creates a CA cert named WebSphereCA.
    - The DIGICERT name is limited to 8 chars, and must be specified.
    - The LABLCERT is case sensitive and can be up to 237 characters.
    - The cert is self-signed. It's a root CA certificate.
    - It's valid until the end of 2010.

As with the keyring name, in CA-Topsecret there are two names one limited to 8 characters and a ‘LABLCERT’ name that is mixed case up to 237 characters.

CA-ACF2 CA Certificates

RACF CA certificate

```
   RACDCERT CERTAUTH GENCERT -
SUBJECTSDN(CN('WAS CertAuth for Security - Domain') -
OU('WebSphere for zOS')) WITHLABEL('WebSphereCA')  TRUST -
NOTAFTER(DATE(2010/12/31))
```

- **The equivalent CA-ACF2 CA Certificate:**
  - GENCERT CERTAUTH SUBJSDN(cn='WASCertAuth for - Security Domain' OU='WebSphere for zOS') - LABEL(WebSphereCA) EXPIRE(12/31/2010)
    - This creates a CA cert named WebSphereCA.
    - The cert is self-signed. It's a root CA certificate.
    - It's valid until the end of 2010.

The CA-ACF2 command is very similar in structure and content to the RACF command.

A RACF Server Certificate

- **A RACF command from BBOWBRAK to create a server Certificate:**
  - RACDCERT ID (MKACRU) GENCERT - SUBJECTSDN(CN('MKACRU.BBOC001') O('IBM') - OU('CB390')) WITHLABEL('DefaultWASCert.BBOC001') - SIGNWITH(CERTAUTH LABEL('WebSphereCA')) - NOTAFTER(DATE(2010/12/31))
    - Creates a cert named DefaultWASCert.BBOC001, owned by user MKACRU.
    - The cert name is case sensitive and can be up to 237 characters.
    - The cert is signed by the RACF cert WebSphereCA.
    - The Common Name (CN) really should be the host name of the server.

This server certificate is one that is created by RACF and signed with the CA certificate that was created earlier in the Security Domain.

It is assigned to the userid of the controller address space.

CA-Top Secret Server Certificate

RACF command:

RACDCERT ID (MKACRU) GENCER- SUBJECTSDN(CN('MKACRU.BBOC001') O('IBM') - OU('CB390')) WITHLABEL('DefaultWASCert.BBOC001') SIGNWITH(CERTAUTH - LABEL('WebSphereCA')) NOTAFTER(DATE(2010/12/31))

- **The equivalent CA-Top Secret Cert:**
  - TSS GENCERT(MKACRU) DIGICERT(MKACRU1) - SUBJECTN('CN="MKACRU.BBOC001" O="IBM" - OU="CB390"') NADATE(12/31/10) - KEYUSAGE('HANDSHAKE') - LABLCERT('DefaultWASCert.BBOC001') - SIGNWITH(CERTAUTH,CA001)
    - This creates a cert named DefaultWASCert.BBOC001, owned by user MKACRU.
    - The DIGICERT name is limited to 8 chars, and must be specified.
    - The LABLCERT is case sensitive and can be up to 237 characters.
    - The cert is signed with the WebSphereCA certificate. The signing cert is specified by its digicert name, not its lablcert name.

The syntax here is pretty much the same as we've seen before.

The 'GENCERT(MKACRU)' creates a certificate for the userid MKACRU.

The certificate is signed with the CA certificate created earlier. Note that the name of the CA certificate referenced in this command is the 8 character 'DIGICERT' name, not the 'LABLCERT' name.

CA-ACF2 Server Certificate

RACF command:

RACDCERT ID (MKACRU) GENCERT SUBJECTSDN(CN('MKACRU.BBOC001') O('IBM') -
OU('CB390')) WITHLABEL('DefaultWASCert.BBOC001') SIGNWITH(CERTAUTH -
LABEL('WebSphereCA')) NOTAFTER(DATE(2010/12/31))

- **The equivalent CA-ACF2 Cert:**
  - GENCERT MKACRU SUBJSDN(CN='MKACRU.BBOC001' -
    O='IBM' OU='CB390') LABEL(DefaultWASCert.BBOC001) -
    SIGNWITH(CERTAUTH LABEL (WebSphereCA)) -
    EXPIRE(12/31/2010)
    - This creates a cert named DefaultWASCert.BBOC001, owned by user MKACRU.
    - The cert is signed with the WebSphereCA certificate.

CA-ACF2 command is very similar to the RACF command.

Connecting RACF Certificates

- **A RACF command from BBOWBRAK to connect a cert to a keyring:**
  - RACDCERT ID(MKACRU) CONNECT -
    (LABEL('DefaultWASCert.BBOC001') -
    RING(WASKeyring) DEFAULT)
    - Connects a cert named DefaultWASCert.BBOC001 to keyring WASKeyring owned by user MKACRU.
    - Makes DefaultWASCert.BBOC001 the default cert in the keyring.

This command connects the servers' certificate to the servers' keyring.

Not shown here, but you would also have to connect the CA's certificate to the this keyring.

Connecting CA-Top Secret Certificates

RACF Command:

```
RACDCERT ID(MKACRU) CONNECT (LABEL('DefaultWASCert.BBOC001') -
      RING(WASKeyring ) DEFAULT)
```

- **The equivalent CA-Top Secret Command:**
  - TSS ADD(MKACRU) KEYRING(MKACRU1) - RINGDATA(MKACRU1,MKACRU1) DEFAULT - USAGE(PERSONAL)
    - Connects a cert with the DIGICERT name of MKACRU1 to the keyring named MKACRU1 owned by user MKACRU.
    - RINGDATA specifies the certificate digicert name and certificate owner.
    - Makes the cert the default cert in the keyring.
  - Now for the equivalent CA-ACF2 command...

In CA-Topsecret you add the certificate to the keyring specifying the 8 character DIGICERT name and the userid of the owner of the keyring. IN the RINGDATA field you specify the 8 character DIGICERT name of the certificate that your are adding and the owner of that certificate.

Connecting CA-ACF2 Certificates

RACF Command:

```
RACDCERT ID(MKACRU) CONNECT (LABEL('DefaultWASCert.BBOC001') -
      RING(WASKeyring ) DEFAULT)
```

- **The equivalent CA-ACF2 Command:**
  - SET PROFILE(USER) DIV(KEYRING)
  - CONNECT CERTDATA(MKACRU) - LABEL(DefaultWASCert.BBOC001) KEYRING(MKACRU) - RINGNAME(WASKeyring) DEFAULT
    - Connects a cert named DefaultWASCert.BBOC001 to keyring WASKeyring owned by user MKACRU.
    - Makes MKACRU.BBOC001 the default cert in the keyring.

- First you tell CA-ACF2 that you are operating against a user profile and its' keyring
- Then you connect the certificate owned by user MKACRU.

WebSphere and CA-Top Secret Issues

- **Issues CA-Top Secret customers have had with WebSphere setup.**
  - No prior experience with SSL, Keyrings and Certs.
  - Customization Dialog output is RACF commands.
    - And how do you convert the commands if you don't know what they do?
  - The 'two names' issue for keyrings and certs.
    - WebSphere assumes names longer than 8 characters.
    - Do I use DIGICERT or LABLCERT?
    - KEYRING or LABLRING?
    - Case sensitivity
  - The same issues RACF customers have:
    - Ownership of certs and keyrings
    - FACILITY class profiles IRR.DIGTCERT.LIST and LISTRING
    - WebSphere diagnostic messages

Probably the biggest issue that we have seen is that security administrators have very little if any experience with keyrings and certificates. This is true in many environments, not only in customers using CA-Topsecret and CA-ACF2.

Converting RACF commands is not an easy task. There isn't any real good documentation with examples.

The fact that CA-Topsecret requires two names for keyrings and certificates is a little confusing, since RACF does not. The question is which one does WebSphere use. What do I specify in the ISPF dialogs?

Since WebSphere is going to look for the keyring name specified in the dialogs, that name must be used as the 'LABLRING' value when defining the keyring.

Keyring name (LABLRING) and certificate name (LABLCERT) are both case sensitive.

WebSphere and CA-ACF2 Issues

- **Issues CA-ACF2 customers have had with WebSphere setup.**
  - No prior experience with SSL, Keyrings and Certs.
  - Customization Dialog output is RACF commands.
    - But the ACF2 commands are very similar.
    - No 'two names' issue for keyrings and certs with ACF2.
  - The same issues RACF customers have:
    - Ownership of certs and keyrings
    - FACILITY class profiles IRR.DIGTCERT.LIST and LISTRING
    - WebSphere diagnostic messages

Fewer issues than at CA-Topsecret customers, although again experience with keyrings and certificates is a problem.

Many customers run into the ownership of the keyring problem.

Remember to define the FACILITY class profiles

- IRR.DIGTCERT.LIST
- IRR.DIGTCERT.LISTRING

And provide the appropriate permissions so that the servers can access their keyrings and certificates.

Troubleshooting Global Security Problems

- **If your Administrative Console doesn't answer...**
  - See if your server(s) came up.
  - Check for SSL problems. (find 'gsk' error messages)
    - gsk return codes indicate problems with RACF keyrings or certs.
    - See 'System SSL Programming', Chapter 13, SC24-5901 for return code meaning.
    - Look in z/OS 1.4 library for SC24-5901-02.
  - Double check BBOxBRAK jobs.
  - ICH408I messages may indicate the server doesn't have access to ICSF CSFSERV class profiles.
  - If ICSF/crypto hardware is available, but WebSphere doesn't have permission, System SSL doesn't use software encryption.

Trouble shooting security problems is never easy.

If after you turn on Global Security the Deployment Manager, or in the case of a Standalone Server the server, doesn’t come up. Which means you can not get to the Admin console:

- The first thing you should look for are error message indicating problems with SSL.
- Error messages from SSL are prefixed with GSK.
  - Look in the output of the controller region.
  - GSK return codes are documented in the System SSL Programming Guide.
- If you are using RACF, you should double check the BBOxRAK jobs.
- Look for ICH408I messages in the syslog
  - These might indicate problems accessing the keyrings or certificates
- If you are using ICSF, make sure there aren’t any error messages from ICSF.

Troubleshooting Global Security Problems

- **If your Administrative Console is available but...**
  - The Nodes will not synch
  - This might indicate that the Node Agents and servers do not agree if security is on or off.
  - Look in the Node Agents' output and the server controllers' output
    - Message: **SECJ0210I:**
      - Security enabled true or Security enabled false
  - This must match with what the Deployment Manager thinks.
    - same message id
  - If the DM thinks security is on and the NA thinks it is off
    - Using the Admin Console turn security off
    - Do not synch to nodes
    - Restart the Deployment Manager
  - Security should now be off and you can start the process of turning it back on.

It's is entirely possible that after you turn Global Security on you can start your Deployment Manager, but it will not communicate with your Node Agents and servers.

This usually indicates that there is something wrong with one of their keyrings or certificates, or they do not agree on the state of Global Security in the cell.

You can look at the output of the control regions and the Node Agents and see if they think security is on or off.

In order for them to successfully communicate, they must agree.

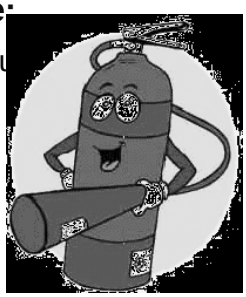
If the Deployment Manager thinks that security is enabled and the rest of the cell does not, this should be fairly easy to fix. This usually means that you can get into the Admin Console, but can not synch the nodes.

You should use the Admin Console to turn Global Security off. When you save the configuration make sure you uncheck the Synchronize to nodes check box.

Now when you restart the Deployment Manager all of the address space should agree and synchronization can proceed.

Deactivating Global Security

- **If the DM thinks security is off and the NA thinks it is on:**
  - Shut down your servers and Node Agents
  - From the bin directory of the Application Server, issue:
    - wsadmin.sh -conntype NONE
  - At the command-line prompt for wsadmin, issue:
    - securityoff
  - Restart the server and the Node Agent.
  - This must be repeated for each node.
- **If you can't log on to the Administrative Console:**
  - From the bin directory of the Deployment Manager, issu
    - wsadmin.sh -conntype NONE
  - At the command-line prompt for wsadmin, issue:
    - securityoff
  - Restart the Deployment Manager.



Once you have determined the problem and have corrected it, you have to get things back in synch.

If the server or Node Agent is not communicating with the Deployment Manager, but you can log into the Admin console:

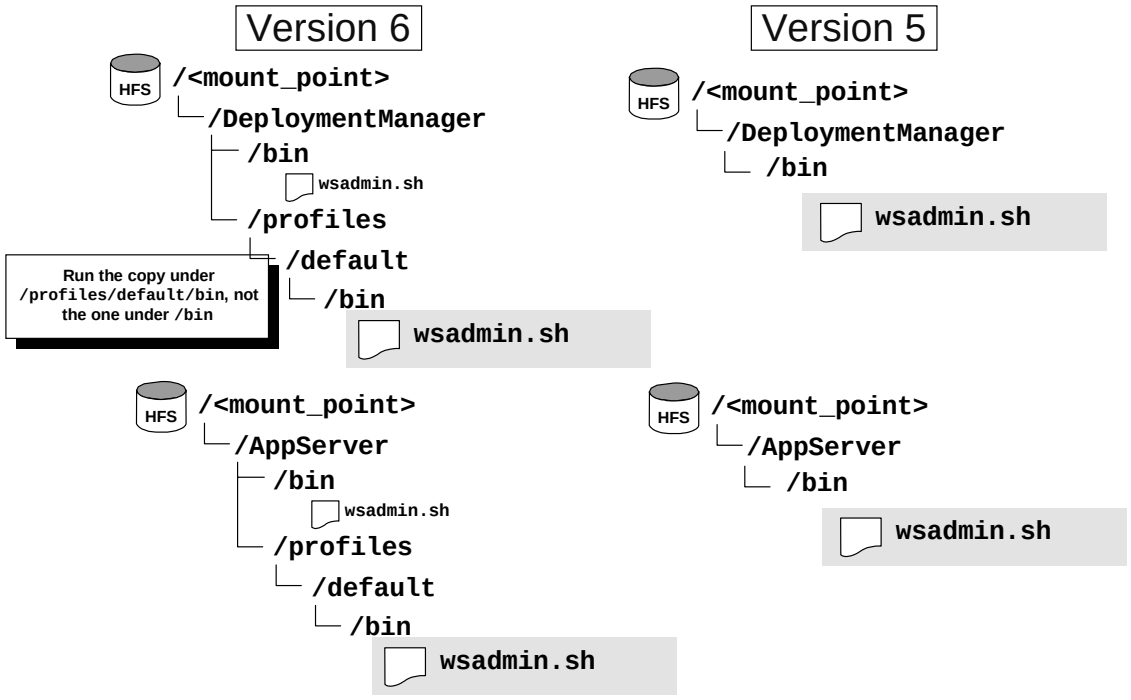
Go to the console and turn off Global Security. You should shut down and restart all of the address spaces in the cell.

The wsadmin.sh shell script must be run in each node, AppServer, Deployment Manager. The script will only turn security off in its' node, so it must be run from each node.

The script is located in /profiles/default/bin directory under AppServer or under DeploymentManager.

The script should be run under the WebSphere administrator userid.

wsadmin Shell Script



In WebSphere V6 there are some new directories in the configuration HFS. Specifically the profiles directory under the AppServer and DeploymentManager directories.

The shell scripts exist in two places in the structure. In order for the wsadmin shell script to find the correct security.xml file the script must be run the right directory. The script uses relative position within the HFS to find the security.xml.

In order for the script to work correctly and in this case find the security.xml file, in WebSphere V6 this is the bin directory located under profiles/default in both the AppServer and DeploymentManager directories.

In Summary...

- **Enabling Global Security forces SSL communication between WebSphere address spaces**
- **WebSphere uses SSL to provide confidentiality between clients and the server.**
- **RACF provides keyrings and certificates in support of SSL.**
- **The Customization dialogs generate the RACF definitions needed.**
- **CA customers have had their share of problems, just like everybody else.**

End of Document