Digital Certificate and RACDCERT – Part 1 and Part 2

SHARE Orlando, FL
Sessions 5534, 5535
February 27th 2008

Wai Choi
IBM Corporation
RACF Development
Poughkeepsie, NY

e-mail: wchoi@us.ibm.com

Trademarks

Agenda

- Overview on RACF Support for Certificates

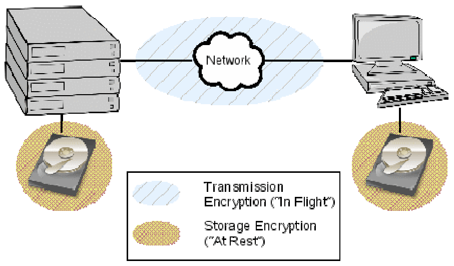
- RACDCERT basics

- Some common questions on certificate management
  1. Using certificate name filtering
  2. Renewing certificate with or without new key pair
  3. Sharing a certificate between two servers

- What's new in RACF Certificate Support on V1R8 and V1R9

What is a digital certificate for?

- To bind a person/entity (subject, owner) to the public key through a Certificate Authority
- Mainly used to secure the data communication between server and client
- Also used to encrypt ‘data at rest’ – mails, data on disk, tapes…


Network
Transmission Encryption ("In Flight")
Storage Encryption ("At Rest")

What's inside a Certificate?

I, [illegible], do hereby certify that [illegible] is who he/she claims to be and that his/her public key is [illegible].
[illegible]
CA's Signature

Certificate Info
- version
- serial number
- signature algorithm ID
- issuer's name
- validity period
- subject's name
- subject's public key
- extensions

Certificate Signature

This is the hash/encrypt algorithm used in the signature

The certificate binds a public key to a subject

CA signs the above cert info by encrypting the hash with its private key

You can NOT change ANY of the certificate information!

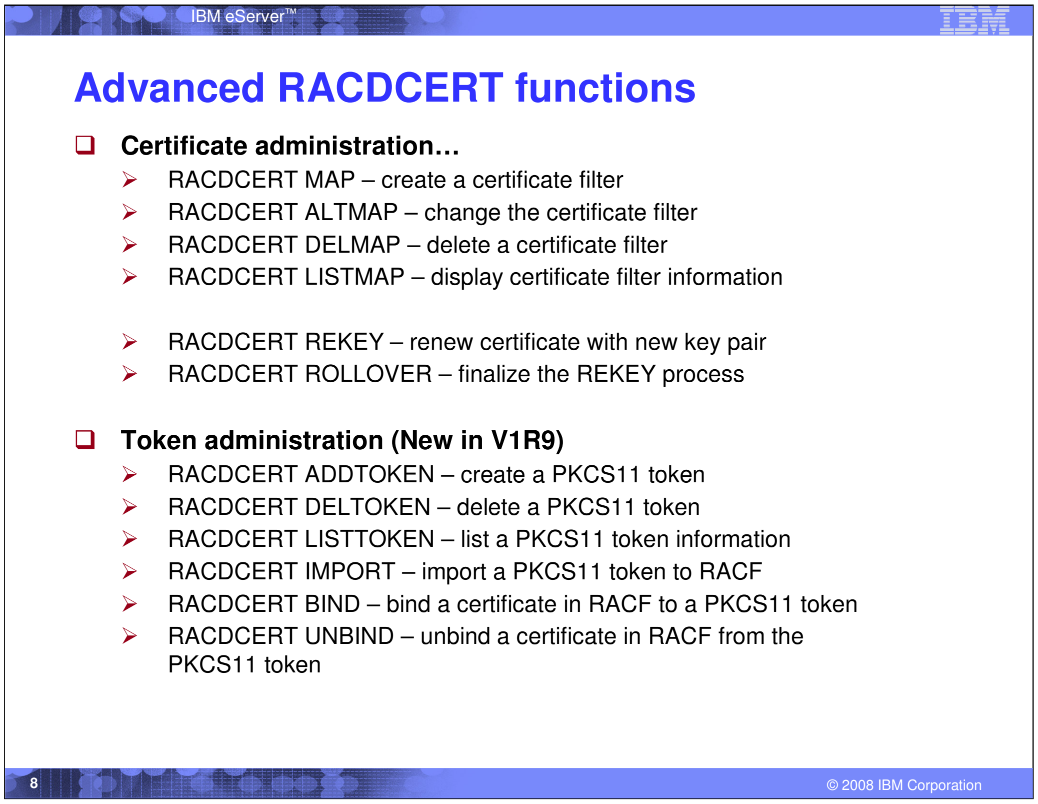
Digital certificate support Overview

- Support through RACF
  - RACDCERT command
    - Read, write functions on certificates, key rings, certificate filters
  - R_Datalib callable services
    - Read functions on certificates in a key ring
    - Called by System SSL APIs which are used by FTP, Telnet…
  - initACEE callable services
    - Using certificate to authenticate to RACF
  - R_PKIServ callable services
    - Interface to call PKI Services
- Support through PKI Services

Basic RACDCERT functions

- **Certificate generation**
  - RACDCERT GENCERT – generate and install a certificate
  - RACDCERT GENREQ – generate a certificate request
- **Certificate installation**
  - RACDCERT ADD – install a certificate / add key to PKDS
- **Certificate administration**
  - RACDCERT ADDRING – create a key ring
  - RACDCERT CONNECT – place a certificate in a key ring
  - RACDCERT REMOVE – remove a certificate from a key ring
  - RACDCERT LISTRING – display key ring information
  - RACDCERT DELRING – delete a key ring

  - RACDCERT LIST – display certificate information from an installed certificate
  - RACDCERT ALTER – change certificate installation information
  - RACDCERT DELETE – delete a certificate from RACF
  - RACDCERT CHECKCERT – display certificate information from a dataset
  - RACDCERT EXPORT – export a certificate


IBM eServer™

Advanced RACDCERT functions

- Certificate administration…
  - RACDCERT MAP – create a certificate filter
  - RACDCERT ALTMAP – change the certificate filter
  - RACDCERT DELMAP – delete a certificate filter
  - RACDCERT LISTMAP – display certificate filter information

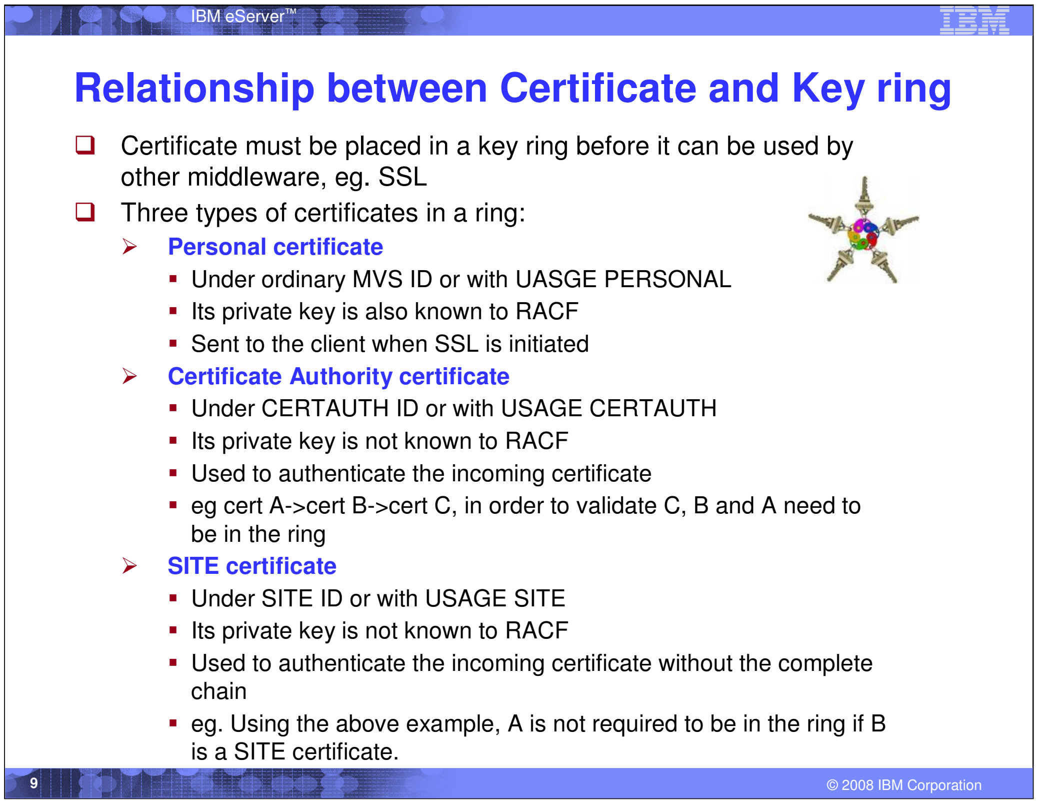
  - RACDCERT REKEY – renew certificate with new key pair
  - RACDCERT ROLLOVER – finalize the REKEY process

- Token administration (New in V1R9)
  - RACDCERT ADDTOKEN – create a PKCS11 token
  - RACDCERT DELTOKEN – delete a PKCS11 token
  - RACDCERT LISTTOKEN – list a PKCS11 token information
  - RACDCERT IMPORT – import a PKCS11 token to RACF
  - RACDCERT BIND – bind a certificate in RACF to a PKCS11 token
  - RACDCERT UNBIND – unbind a certificate in RACF from the PKCS11 token

8 © 2008 IBM Corporation

•PKCS#11 is the Cryptographic Token Interface Standard which specifies an application programming interface (API) to devices (tokens) which hold cryptographic information and perform cryptographic functions.

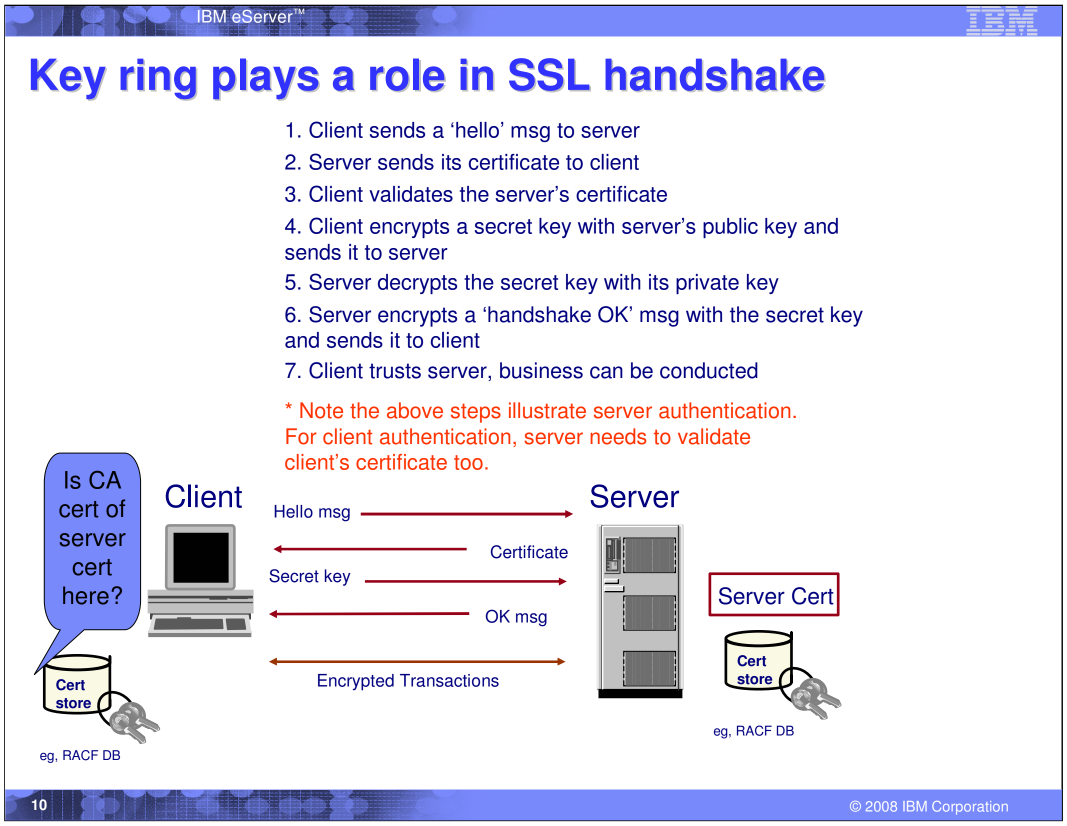
•On most single user systems a token is an actual smart card. On z/OS tokens will be virtual, conceptually similar to RACF key rings



Relationship between Certificate and Key ring

- Certificate must be placed in a key ring before it can be used by other middleware, eg. SSL
- Three types of certificates in a ring:
  - **Personal certificate**
    - Under ordinary MVS ID or with UASGE PERSONAL
    - Its private key is also known to RACF
    - Sent to the client when SSL is initiated
  - **Certificate Authority certificate**
    - Under CERTAUTH ID or with USAGE CERTAUTH
    - Its private key is not known to RACF
    - Used to authenticate the incoming certificate
    - eg cert A->cert B->cert C, in order to validate C, B and A need to be in the ring
  - **SITE certificate**
    - Under SITE ID or with USAGE SITE
    - Its private key is not known to RACF
    - Used to authenticate the incoming certificate without the complete chain
    - eg. Using the above example, A is not required to be in the ring if B is a SITE certificate.

•A certificate can be connected to different key rings, playing different roles.
•The certificate type can be overridden by the USAGE keyword in the CONNECT command.



Key ring plays a role in SSL handshake

1. Client sends a 'hello' msg to server
2. Server sends its certificate to client
3. Client validates the server's certificate
4. Client encrypts a secret key with server's public key and sends it to server
5. Server decrypts the secret key with its private key
6. Server encrypts a 'handshake OK' msg with the secret key and sends it to client
7. Client trusts server, business can be conducted

* Note the above steps illustrate server authentication. For client authentication, server needs to validate client's certificate too.

•Validation involves checking the CA signature on the certificate

•Need CA's public key to validate

•CA's public key is on the CA's certificate

•So need the presence of the CA's certificate

A Quiz

Given:

- CA-S is the CA cert which signed the server cert S
- CA-C is the CA cert which signed the client cert C
- Sring is the server's key ring, Cring is the client's key ring

Questions:

- What cert(s) needed in Sring? in Cring?
- Usage in ring - Certauth (*A*) or Personal (*P*)?
- Which cert in which ring do you need to have its private key?

1. **For Server authentication**

   Ans: Ring-server: CA-S (*A*), **S** (*P*)   Ring-client: CA-S (*A*)
2. **For Client authentication (implies server authentication too)**

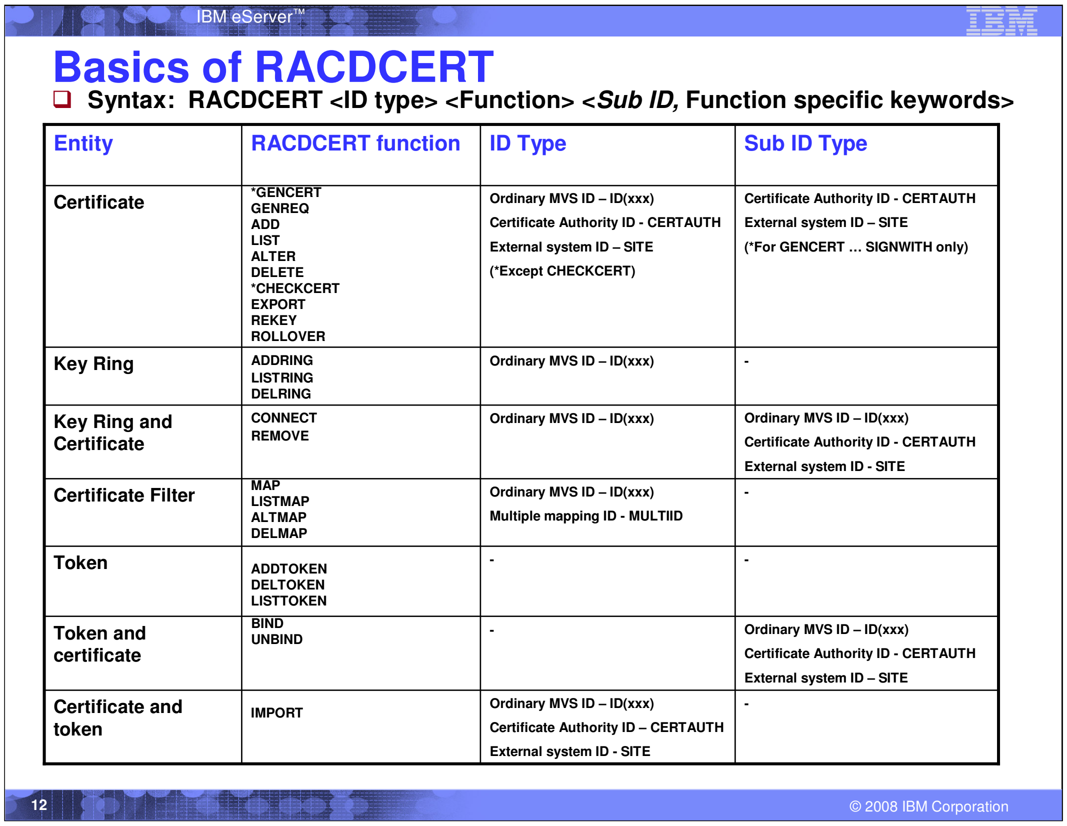
   Ans: Ring-server: CA-S (*A*), **S** (*P*), CA-C (*A*)   Ring-client: CA-C (*A*), **C** (*P*), CA-S (*A*)

Further thinking:

3. **Would it be simpler (for which case?) if both the server and client certs were signed by the same CA cert, say CA-S? How would the rings look like?**

   Ans: Client authentication will be simpler

   Ring-server: CA-S (*A*), **S** (*P*)   Ring-client: CA-S (*A*), **C** (*P*)



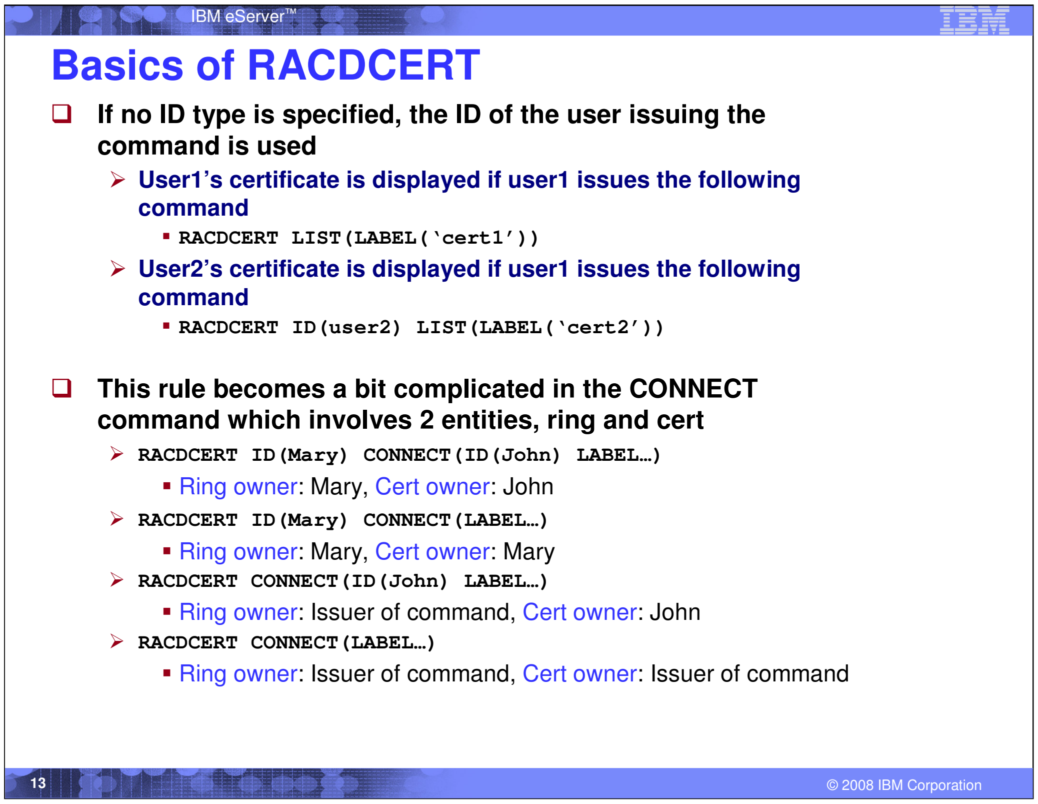
Basics of RACDCERT

- Syntax: RACDCERT <ID type> <Function> <*Sub ID,* Function specific keywords>

Entity	RACDCERT function	ID Type	Sub ID Type
Certificate	*GENCERT GENREQ ADD LIST ALTER DELETE *CHECKCERT EXPORT REKEY ROLLOVER	Ordinary MVS ID – ID(xxx) Certificate Authority ID - CERTAUTH External system ID – SITE (*Except CHECKCERT)	Certificate Authority ID - CERTAUTH External system ID – SITE (*For GENCERT … SIGNWITH only)
Key Ring	ADDRING LISTRING DELRING	Ordinary MVS ID – ID(xxx)	-
Key Ring and Certificate	CONNECT REMOVE	Ordinary MVS ID – ID(xxx)	Ordinary MVS ID – ID(xxx) Certificate Authority ID - CERTAUTH External system ID - SITE
Certificate Filter	MAP LISTMAP ALTMAP DELMAP	Ordinary MVS ID – ID(xxx) Multiple mapping ID - MULTIID	-
Token	ADDTOKEN DELTOKEN LISTTOKEN	-	-
Token and certificate	BIND UNBIND	-	Ordinary MVS ID – ID(xxx) Certificate Authority ID - CERTAUTH External system ID – SITE
Certificate and token	IMPORT	Ordinary MVS ID – ID(xxx) Certificate Authority ID – CERTAUTH External system ID - SITE	-

•This slide just indicates the overall syntax in a high level format to illustrate the basic concepts involved. For detailed syntax, refer to the Command Language Reference.

•Token is a special entity. It is not a RACF entity. It belongs to ICSF. RACDCERT provides a link between the certificate in RACF and the token in ICSF.

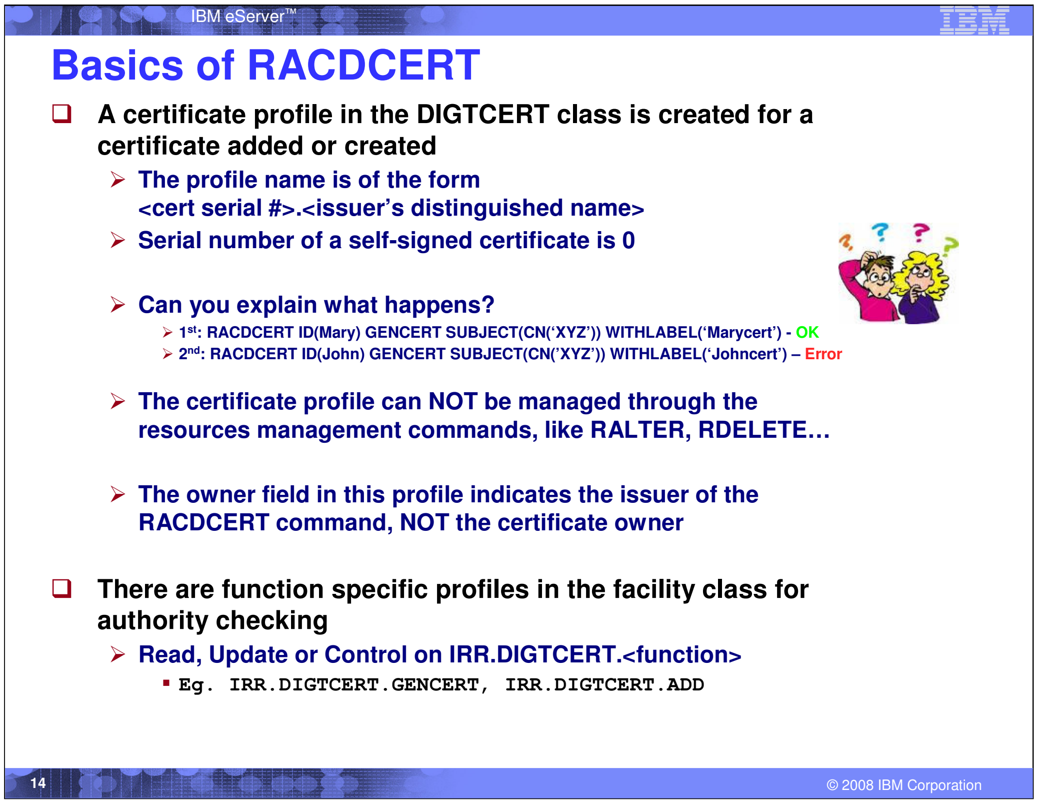


Basics of RACDCERT

- **If no ID type is specified, the ID of the user issuing the command is used**
  - **User1's certificate is displayed if user1 issues the following command**
    - `RACDCERT LIST(LABEL('cert1'))`
  - **User2's certificate is displayed if user1 issues the following command**
    - `RACDCERT ID(user2) LIST(LABEL('cert2'))`

- **This rule becomes a bit complicated in the CONNECT command which involves 2 entities, ring and cert**
  - `RACDCERT ID(Mary) CONNECT(ID(John) LABEL…)`
    - Ring owner: Mary, Cert owner: John
  - `RACDCERT ID(Mary) CONNECT(LABEL…)`
    - Ring owner: Mary, Cert owner: Mary
  - `RACDCERT CONNECT(ID(John) LABEL…)`
    - Ring owner: Issuer of command, Cert owner: John
  - `RACDCERT CONNECT(LABEL…)`
    - Ring owner: Issuer of command, Cert owner: Issuer of command

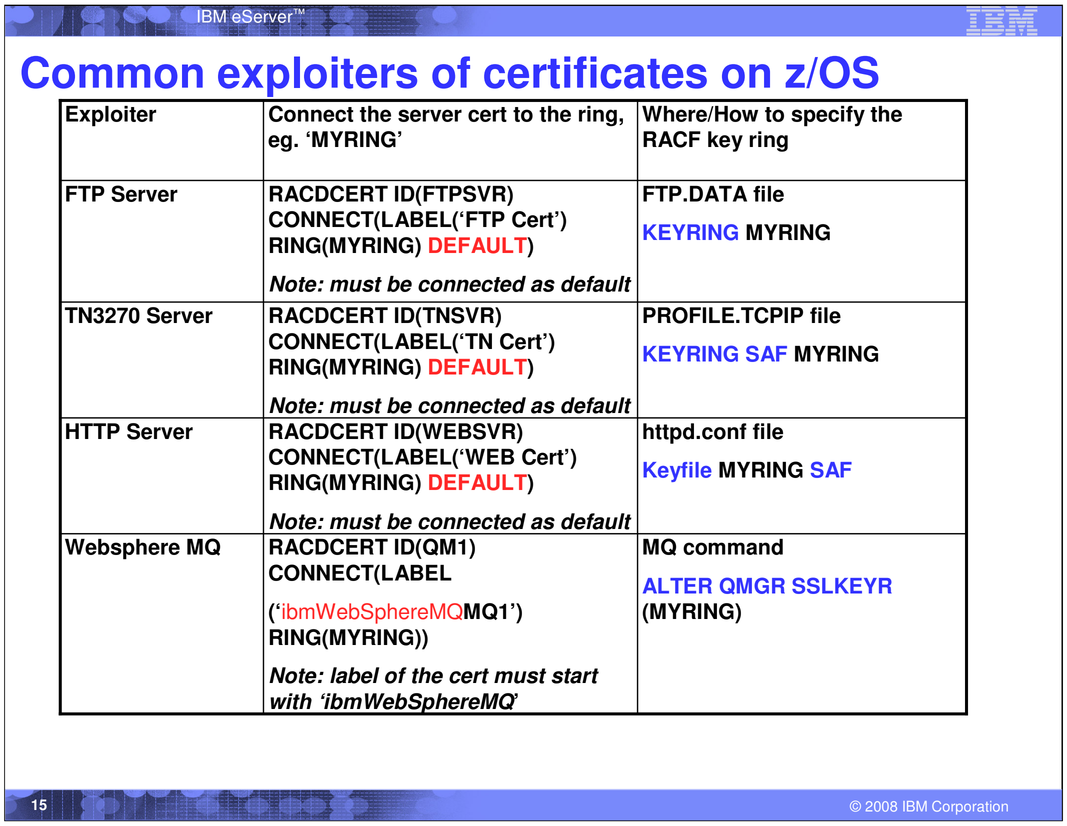
Label is case sensitive, max length is 32 characters



Basics of RACDCERT

- A certificate profile in the DIGTCERT class is created for a certificate added or created
  - The profile name is of the form <cert serial #>.<issuer's distinguished name>
  - Serial number of a self-signed certificate is 0
  - Can you explain what happens?
    - 1st: RACDCERT ID(Mary) GENCERT SUBJECT(CN('XYZ')) WITHLABEL('Marycert') - OK
    - 2nd: RACDCERT ID(John) GENCERT SUBJECT(CN('XYZ')) WITHLABEL('Johncert') – Error
  - The certificate profile can NOT be managed through the resources management commands, like RALTER, RDELETE…
  - The owner field in this profile indicates the issuer of the RACDCERT command, NOT the certificate owner
- There are function specific profiles in the facility class for authority checking
  - Read, Update or Control on IRR.DIGTCERT.<function>
    - `Eg. IRR.DIGTCERT.GENCERT, IRR.DIGTCERT.ADD`

•If you try to re-generate a self-signed certificate with the same subject distinguished name without deleting the previous one, you will get an error saying that the profile is already defined, EVEN if you generate it under a different user id.
•This is because in a self-signed case, the issuer's distinguished name is the same as the subject's distinguished name, say CN=xyz. The profile created for the self-signed certificate is 00.CN=xyz.
•The signing certificate assigns serial numbers to the certificates it issued and it keeps track of these issued numbers so that there will not be any duplicates.



Common exploiters of certificates on z/OS

Exploiter	Connect the server cert to the ring, eg. 'MYRING'	Where/How to specify the RACF key ring
FTP Server	RACDCERT ID(FTPSVR) CONNECT(LABEL('FTP Cert') RING(MYRING) DEFAULT) *Note: must be connected as default*	FTP.DATA file KEYRING MYRING
TN3270 Server	RACDCERT ID(TNSVR) CONNECT(LABEL('TN Cert') RING(MYRING) DEFAULT) *Note: must be connected as default*	PROFILE.TCPIP file KEYRING SAF MYRING
HTTP Server	RACDCERT ID(WEBSVR) CONNECT(LABEL('WEB Cert') RING(MYRING) DEFAULT) *Note: must be connected as default*	httpd.conf file Keyfile MYRING SAF
Websphere MQ	RACDCERT ID(QM1) CONNECT(LABEL ('ibmWebSphereMQMQ1') RING(MYRING)) *Note: label of the cert must start with 'ibmWebSphereMQ'*	MQ command ALTER QMGR SSLKEYR (MYRING)

The certificates are assumed to be created under a personal ID. The usage is thus implied to be PERSONAL.

A certificate represents a RACF user

- User can be identified to RACF through certificate if his certificate is in the RACF DB
- If there are thousands of users, thousands of certificates need to be installed…

More Sophisticated Certificate Support from RACF

- Solution 1: Certificate Name Filtering
  - Supported by RACF
- Solution 2: HostIdMapping
  - Supported by RACF and PKI Services
- No user certificate needs to be installed.

More Sophisticated Support from RACF…

- **Certificate Name Filtering – RACDCERT MAP**
  - **Definition of a set of rules ('filters') based on the subject's or issuer's distinguished names (or both)**
  - **Can map one or more certificates to a filter**
  - **Need to raclist DIGTNMAP class**

  - **Examples:**
    - ► **Create a filter to associate ID VUSER to any user presenting a certificate issued by Verisign Class 1 Individual Subscriber**

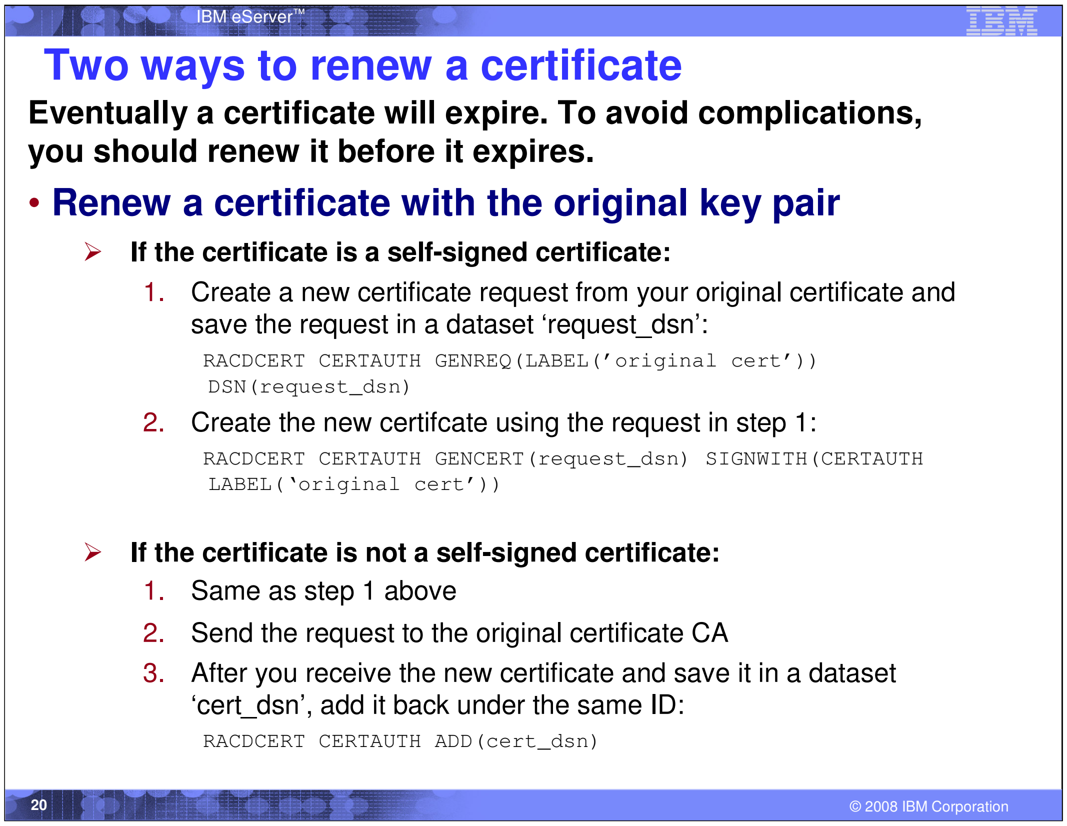
```
RACDCERT MAP ID(VUSER) IDNFILTER('OU=Verisign Class 1
Individual Subscriber.O=Verisign, Inc.L=Internet')…
```

    - ► **Create a filter to associate ID RACFGP to any user presenting a certificate with subject's distinguished name OU=RACF.O=IBM**

```
RACDCERT MAP ID(RACFGP) SDNFILTER ('OU=RACF.O=IBM')…
```

More Sophisticated Support from RACF…

- **HostIdMapping**
  - A client can present a certificate containing a HostIdMapping extension to the server
  - This extension contains a host name and a subject id, eg. use1@abc.com
  - RACF will honor this extension if
    - the issuing CA cert is marked HIGHTRUST
    - the host name in the extension matches a profile IRR.HOST.<host name> in the SERVAUTH class under the subject id, eg. user1 has access to IRR.HOST.abc.com
  - RACDCERT can't create this extension, PKI Services can



Two ways to renew a certificate

Eventually a certificate will expire. To avoid complications, you should renew it before it expires.

- **Renew a certificate with the original key pair**

  - **If the certificate is a self-signed certificate:**
    1. Create a new certificate request from your original certificate and save the request in a dataset 'request_dsn':
       ```
       RACDCERT CERTAUTH GENREQ(LABEL('original cert'))
        DSN(request_dsn)
       ```
    2. Create the new certifcate using the request in step 1:
       ```
       RACDCERT CERTAUTH GENCERT(request_dsn) SIGNWITH(CERTAUTH
        LABEL('original cert'))
       ```

  - **If the certificate is not a self-signed certificate:**
    1. Same as step 1 above
    2. Send the request to the original certificate CA
    3. After you receive the new certificate and save it in a dataset 'cert_dsn', add it back under the same ID:
       ```
       RACDCERT CERTAUTH ADD(cert_dsn)
       ```

If the self-signed certificate is under a personal ID, step 2 has to be issued from that ID since SIGNWITH can not accept a person ID. Without specifying CERTAUTH or SITE, SIGNWITH defaults to the command issuer. So the steps are:
RACDCERT ID(Mary) GENREQ(LABEL('original cert')) DSN(request_dsn)
RACDCERT ID(Mary) GENCERT(request_dsn)
SIGNWITH(LABEL('original cert'))

Two ways to renew a certificate…

- **Renew a certificate with a new key pair**

  The longer a key pair is used, the more likely it is to be cracked. The key pair should be periodically changed. Two RACDCERT functions are provided:

  ➢**RACDCERT REKEY**

  -Make a self-signed copy of the original certificate with a new public-private key pair

  ➢**RACDCERT ROLLOVER**

  -Finalize the REKEY operation

  ❖Private key of the old certificate is deleted so that it may not be used again for signing or encryption

  ❖Cert with usage PERSONAL: all keyring occurrences of the old certificate will be replaced with the new one

  ❖Cert with usage CERTAUTH or SITE: the new cert will be added to all keyring occurrences of the old one

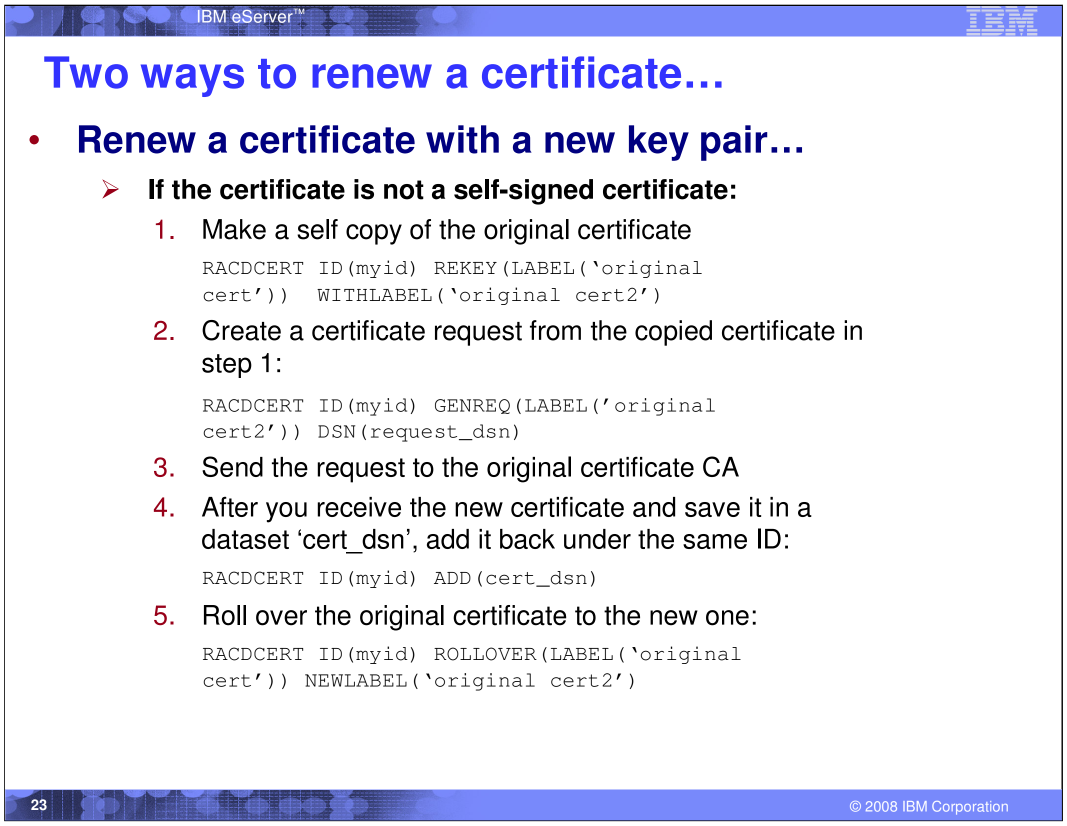
Two ways to renew a certificate…

- **Renew a certificate with a new key pair…**
  - **If the certificate is a self-signed certificate:**
    1. Make a self copy of the original certificate:

       ```
       RACDCERT CERTAUTH REKEY(LABEL(‘original
       cert’))WITHLABEL(‘original cert2’)
       ```

    2. Roll over the original certificate to the new one:

       ```
       RACDCERT CERTAUTH ROLLOVER(LABEL(‘original cert’))
       NEWLABEL(‘original cert2’)
       ```



Two ways to renew a certificate…

- **Renew a certificate with a new key pair…**

  ➢ **If the certificate is not a self-signed certificate:**

  1. Make a self copy of the original certificate
     ```
     RACDCERT ID(myid) REKEY(LABEL(‘original
     cert’))  WITHLABEL(‘original cert2’)
     ```
  2. Create a certificate request from the copied certificate in step 1:
     ```
     RACDCERT ID(myid) GENREQ(LABEL(’original
     cert2’)) DSN(request_dsn)
     ```
  3. Send the request to the original certificate CA
  4. After you receive the new certificate and save it in a dataset ‘cert_dsn’, add it back under the same ID:
     ```
     RACDCERT ID(myid) ADD(cert_dsn)
     ```
  5. Roll over the original certificate to the new one:
     ```
     RACDCERT ID(myid) ROLLOVER(LABEL(‘original
     cert’)) NEWLABEL(‘original cert2’)
     ```

Share keyring, certificate, private key?

- You may share a keyring and hence its content (certificates) amongst different IDs by giving permission to IRR.DIGTCERT.LISTRING
  - Eg. Set up client A, B, C…for server authentication

  (remember the quiz: no private key is involved in the client side)

- Sharing private key is not recommended, but in case you really want to …eg. Avoid buying a separate certificate for another server or client, there is a way

Share a private key between ID SRV1 and SRV2

- Create a keyring under one ID, say SRV1
  - ➢ `RACDCERT ID(SRV1) ADDRING(ShareRing)`

- Create a certificate under CERTAUTH or SITE, not a personal ID
  - ➢ `RACDCERT SITE GENCERT... WITHLABEL('Share Cert')`

- Connect the cert to this ring
  - ➢
```
RACDCERT ID(SRV1) CONNECT(SITE LABEL('Share Cert')
RING(ShareRing) USAGE(PERSONAL) DEFAULT)
```

- Permit both IDs to use this ring
  - ➢
```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ)
ID(SRV1)
```
  - ➢
```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY)
ACCESS(UPDATE) ID(SRV2)
```

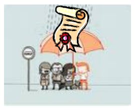
Share a private key between ID SRV1 and SRV2…

- Permit both IDs to use this private key
  - ➢ `RDEF FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)`
  - ➢ `PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ACCESS(CONTROL) ID(SRV1 SRV2)`

*!!!Note: When you share private key as described above, you are allowing the two IDs to access **any** private keys that are stored under SITE or CERTAUTH.*

- There is a better solution in V1R9 that will be discussed later

Share just the certificates between multiple clients for server authentication

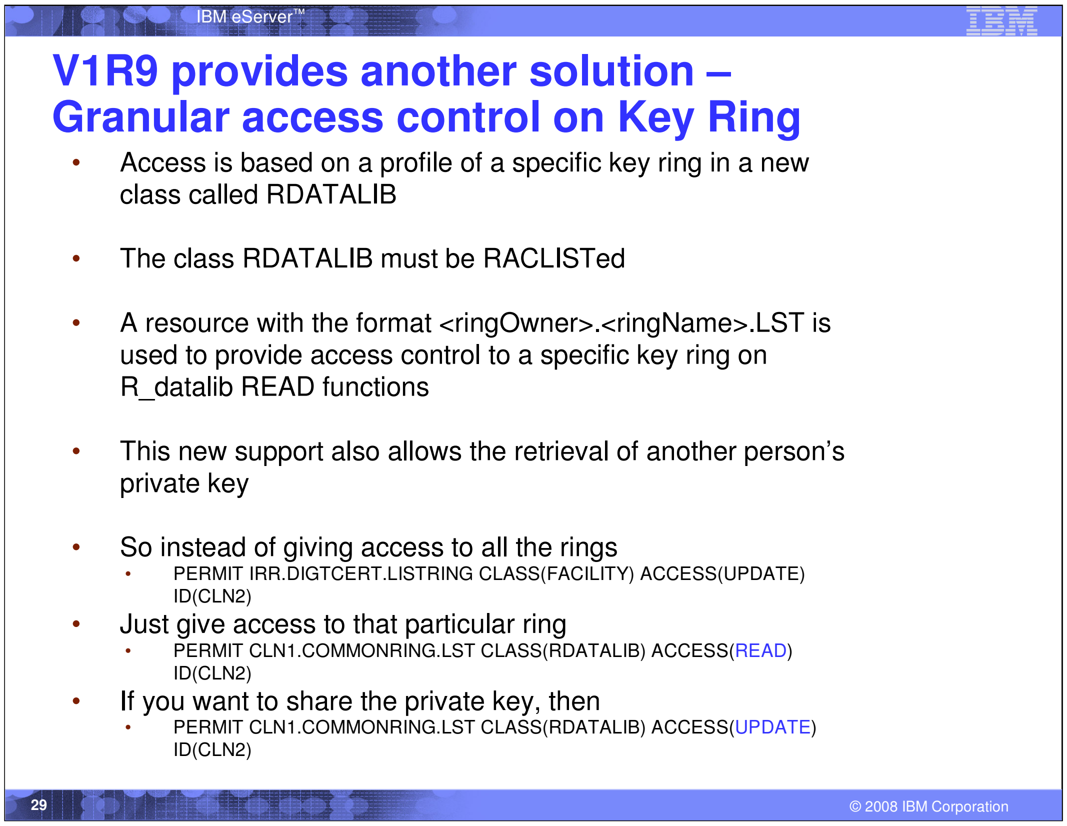


- Create a keyring under one ID, say CLN1
  - `RACDCERT ID(CLN1) ADDRING(CommonRing)`
- Connect all the CA certificates to this ring
  - `RACDCERT ID(CLN1) CONNECT(CERTAUTH LABEL('VeriSign Cert') RING(CommonRing))`
  - `RACDCERT ID(CLN1) CONNECT(CERTAUTH LABEL('GeoTrust Cert') RING(CommonRing))`
  - …
- Permit both IDs to use this ring
  - `PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(CLN1)`
  - `PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(UPDATE) ID(CLN2)`

- If you don't want to share, you need to create a separate ring and connect the same CA certificates for CLN2

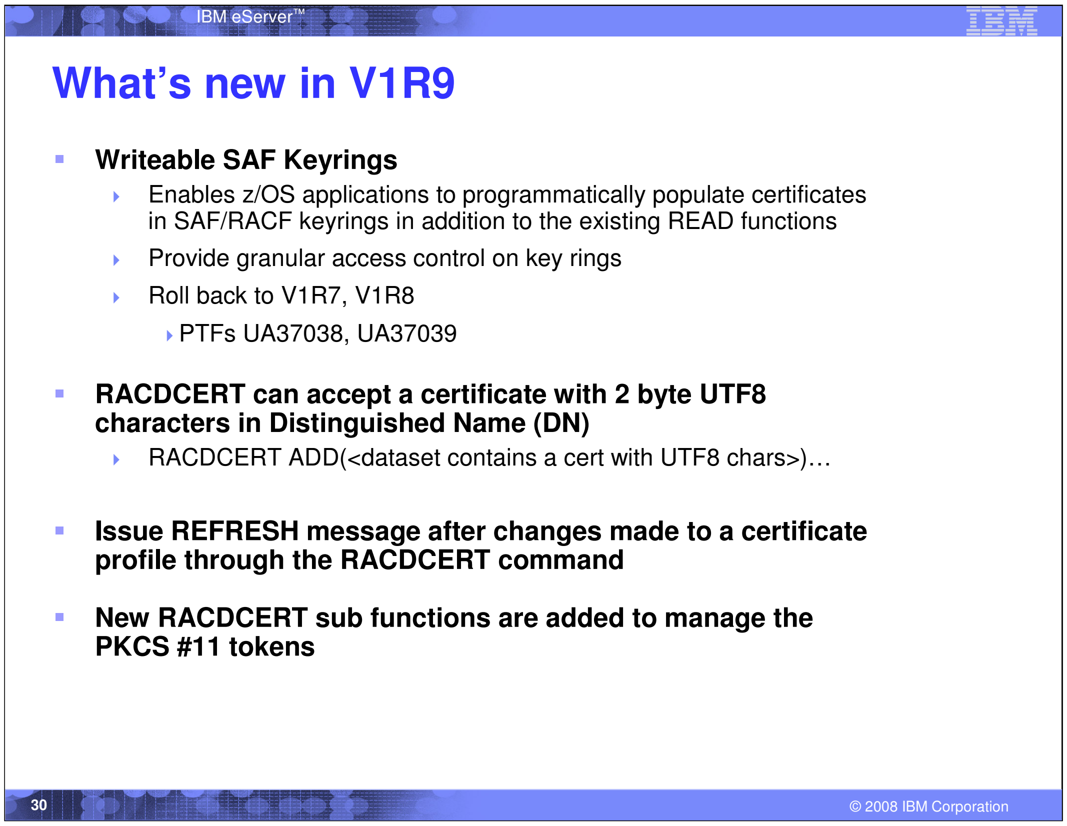
V1R8 provides a better solution – Virtual Key Ring

- All the certificates under a RACF user ID are considered ‘connected’ to a virtual key ring automatically
- Eliminate the work to create key rings and connect certificates to them for all the clients which use the same set of certificates for validation
- This solution is useful for client side SSL applications that don’t do client authentication, for example, multiple FTP clients talking to the same server



V1R9 provides another solution – Granular access control on Key Ring

- Access is based on a profile of a specific key ring in a new class called RDATALIB
- The class RDATALIB must be RACLISTed
- A resource with the format <ringOwner>.<ringName>.LST is used to provide access control to a specific key ring on R_datalib READ functions
- This new support also allows the retrieval of another person's private key
- So instead of giving access to all the rings
  - PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(UPDATE) ID(CLN2)
- Just give access to that particular ring
  - PERMIT CLN1.COMMONRING.LST CLASS(RDATALIB) ACCESS(READ) ID(CLN2)
- If you want to share the private key, then
  - PERMIT CLN1.COMMONRING.LST CLASS(RDATALIB) ACCESS(UPDATE) ID(CLN2)



What's new in V1R9

- **Writeable SAF Keyrings**
  - Enables z/OS applications to programmatically populate certificates in SAF/RACF keyrings in addition to the existing READ functions
  - Provide granular access control on key rings
  - Roll back to V1R7, V1R8
    - PTFs UA37038, UA37039
- **RACDCERT can accept a certificate with 2 byte UTF8 characters in Distinguished Name (DN)**
  - RACDCERT ADD(<dataset contains a cert with UTF8 chars>)…
- **Issue REFRESH message after changes made to a certificate profile through the RACDCERT command**
- **New RACDCERT sub functions are added to manage the PKCS #11 tokens**

•PKCS#11 is the Cryptographic Token Interface Standard which specifies an application programming interface (API) to devices (tokens) which hold cryptographic information and perform cryptographic functions.

•On most single user systems a token is an actual smart card. On z/OS tokens will be virtual, conceptually similar to RACF key rings

References

- **IBM Education Assistant web site:** NEW!

http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp

- **RACF web site:**

http://www.ibm.com/servers/eserver/zseries/zos/racf

- **IBM Redbooks**
  - ►z/OS V1 R8 RACF Implementation (SG24-7248) NEW!
- **Security Server Manuals:**
  - ►RACF Command Language Reference (SC28-1919)
  - ►RACF Security Administrator's Guide (SC28-1915)
  - ►RACF Callable Services Guide (SC28-1921)
  - ►LDAP Administration and Use (SC24-5923)
- **Other Sources:**
  - ►PKIX - http://www.ietf.org/html.charters/pkix-charter.html

Questions???

Appendix

Some common RACDCERT commands

RACDCERT Examples

- **RACDCERT GENCERT:**
  - ► Create a self signed certificate and public-private key pair using ICSF for a CA -- CERTAUTH represents the ‘ID’ of a CA, no ‘SIGNWITH’ needed for self signed cert

```
RACDCERT CERTAUTH GENCERT SUBJECT(...) ICSF…
```

  - ► Create a certificate and public-private key pair using PCICC for ID WEBSRV, signed with a CA certificate named ‘theCA cert’

```
RACDCERT ID(WEBSRV) GENCERT SUBJECT(…) PCICC
SIGNWITH(CERTAUTH LABEL(‘theCA cert’)…
```

  - ► Create a certificate based on a certificate request specified in the dataset ‘CERTREQ.B64’ for ID MYID – you get the request from other system and you want your local CA to sign it. This generates certificate only, no key pair is created.

```
RACDCERT ID(MYID) GENCERT(‘CERTREQ.B64’)
SIGNWITH(CERTAUTH LABEL(‘theCA cert’)…
```

- **RACDCERT GENREQ:**
  - ► Generate a request based on an existing certificate named ‘My Self Signed Cert’ in RACF for ID OUTSRV and put the request in the data set ‘CERTREQ.B64’

```
RACDCERT ID(OUTSRV) GENREQ(LABEL(‘My Self Signed Cert'))
DSN(CERTREQ.B64)…
```

RACDCERT Examples…

- **RACDCERT ADD:**
  - Add a trusted CA certificate under ID CERTAUTH

    ```
    RACDCERT CERTAUTH ADD(‘dataset containing CA cert’) TRUST…
    ```

  - Add a trusted peer certificate under ID SITE

    ```
    RACDCERT SITE ADD(‘dataset containing site cert’) TRUST…
    ```

  - Add a certificate package that contains the private key under ID myid

    ```
    RACDCERT ID(myid) ADD(‘dataset containing my cert’)
    password(‘secret’)…
    ```

  - Replace a previous self signed certificate under ID OUTSRV

    ```
    RACDCERT ID(OUTSRV) ADD(‘dataset containing a CA signed
    cert issued for the request based on the self signed cert’)
    TRUST
    ```

    - GENCERT a self signed cert
    - GENREQ based on the self signed cert
    - Send request to CA and get back a new cert
    - ADD the new cert back under the same ID

RACDCERT Examples…

- **RACDCERT ADDRING:**
  - ► Create a key ring called ‘SSLring’ for ID WEBSRV

```
RACDCERT ID(WEBSRV) ADDRING(SSLring)
```

- **RACDCERT CONNECT:**
  - ► Connect WEBSRV’s own certificate called ‘SSL cert’ to WEBSRV’s key ring called SSLring

```
RACDCERT ID(WEBSRV) CONNECT(LABEL(‘SSL cert’) RING(SSLring)
DEFAULT…)
```

- **RACDCERT MAP:**
  - ► Create a filter to associate ID VUSER to any user presenting a certificate issued by Verisign™ Class 1 Individual Subscriber

```
RACDCERT MAP ID(VUSER) IDNFILTER(‘OU=Verisign Class 1
Individual Subscriber.O=Verisign, Inc.L=Internet’)…
```

  - ► Create a filter to associate ID RACFGP to any user presenting a certificate with subject’s distinguished name OU=RACF.O=IBM

```
RACDCERT MAP ID(RACFGP) SDNFILTER (‘OU=RACF.O=IBM’)…
```

Disclaimer