

Getting Started with Digital Certificates Part II (RACDCERT)



© 2006 IBM Corporation

1

This presentation will guide you through the RACF's world of Digital Certificates. We will attempt to explain how they can be used and how RACF can aid in creating, maintaining, storing and managing your Digital Certificates. We will go through some examples and explanations of the functions of the RACF's RACDCERT command. For an explanation of Symmetric keys, Asymmetric keys and the foundations of digital certificates, refer to Part I of this series, 'Getting Started with Digital Certificates, Part I'.

The Objectives of this presentation

- ❑ To introduce the RACF support for digital certificates through the RACDCERT command
- ❑ To clarify some misunderstandings in the RACDCERT command
- ❑ To discuss some common certificate operations

- The objectives of this presentation is to introduce RACF's support.
- Digital certificates will allow you to have a unique method of identifying the user. Each certificate has certain information regarding the user, certified by the person or organization that issued the certificate.
- The purpose of this module will help you learn how to create, use and manage Digital Certificates and key rings by using the RACDCERT command in RACF.
- The RACDCERT command is a powerful command with many different options. These options can be confusing and we will help you understand which options will best benefit your needs. Once you have these certificates created, how can you use them? We will discuss some of these operations.

Legal Disclaimer

Copyright © 2006 by International Business Machines Corporation.

No part of this document may be reproduced or transmitted in any form without written permission from IBM Corporation.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This information could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or programs(s) at any time without notice.

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

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead. It is the user's responsibility to evaluate and verify the operation of any non-IBM product, program or service.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. IBM is not responsible for the performance or interoperability of any non-IBM products discussed herein.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

IBM Systems™ IBM Education Assistant IBM

Trademarks

- See <http://www.ibm.com/legal/copytrade.shtml> for a list of trademarks.

© 2006 IBM Corporation 4

For a list of trademarks, follow this link.

Basic Rules of RACDCERT

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

Entity	RACDCERT function	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
Key Ring	ADDRING LISTRING DELRING CONNECT REMOVE	Ordinary MVS ID – ID(xxx)
Certificate Filter	MAP LISTMAP ALTMAP DELMAP	Ordinary MVS ID – ID(xxx) Multiple mapping ID - MULTIID

- The RACDCERT command has many functions with function specific keywords for the management of certificates, certificate key rings and certificate mappings.
- For certificates, ID Type is who or what the certificate is to be associated with. The valid ID types would be an ordinary MVS userID (ID), a Certificate Authority (CERTAUTH) or an external system ID (SITE).
- For certificate key rings, the ID type would be an ordinary MVS userID for the ownership of a key ring.
- For certificate filters, the ID type would again be an ordinary MVS userID or a system defined ID (MULTIID) and there is also the ability to map multiple certificates with specific criteria to one userID. Mappings will be discussed in Part III.
- This presentation just indicates the overall syntax in a high level format to illustrate the basic concepts involved. For detailed syntax, refer to the RACF Command Language Reference publication.

Basic Rules 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('cert1'))`
- ❑ To ease certificate management and identification, certificates have labels.
- ❑ There are function specific profiles in the facility class for authority checking
 - Read, Update or Control on `IRR.DIGTCERT.<function>`
 - for example, `IRR.DIGTCERT.GENCERT`, `IRR.DIGTCERT.ADD`

•There are three categories of certificates; Personal (ID), Certificate Authority (CERTAUTH) and SITE. For most certificate operations, one of these three can be used. If none are used, the default would be a Personal certificate (ID) and the userID of the command issuer would be the one used. [enter]

•For instance, if USER1 issues RACDCERT LIST, as the command issuer, the ID would default to USER1 and if USER1 has any certificates, they will be displayed. [enter]

•If USER1 has the authority to list other userID's certificates for example USER2, USER1 can display them by issuing RACDCERT ID(USER2).


•To be able to manage certificates, certificates have labels. Labels are case sensitive, quoted strings with a maximum length of 32 characters. [enter]

•For users to have the RACF authority to issue the RACDCERT command, profiles in the FACILITY class as `IRR.DIGTCERT.<function>` must be created for the specific functions. For example, if USER1 has the need to LIST certificates for other userID's, they would need at least UPDATE authority in the profile `IRR.DIGTCERT.LIST` in the FACILITY class.

Basic Rules 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>**
 - **Unlike the other profiles, the certificate profile can not be managed through the resources management commands.**
 - **The owner field in this profile indicates the issuer of the RACDCERT command, NOT the certificate owner**

- For RACF to manage the certificate that is created or added in the RACF database, a profile is created in the DIGTCERT class. This profile is in the form of certificate serial number dot issuer's distinguished name. As we mentioned previously, each certificate needs to be uniquely identified. That is done with the guarantee that no two certificates will have the combination of the same serial number and the same issuer's distinguished name.
- When RACF generates a new self-signed certificate, serial number 0 is assigned to it. 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 issues and it keeps track of these issued numbers so that there will not be any duplicates.
- Since these certificate profiles need to be managed differently, the resources management commands such as RDEFINE, RALTER and RDELETE cannot be used. Additionally, the owner field in these profiles indicates the issuer of the original RACDCERT command that created the profile, not the owner of the certificate.



IBM Systems™

IBM Education Assistant



Certificate generation – RACDCERT GENCERT, GENREQ

© 2006 IBM Corporation

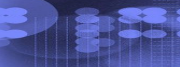
8

Now let's see how we can create certificates and certificate requests.

RACDCERT GENCERT


- **GENerates a CERTificate, and optionally public-private key**
- **Syntax: RACDCERT GENCERT <dataset contains a request> <cert info> <cert label> <trust status> <way to generate key pair> <label of signing cert>**
- **Can utilize z/OS hardware crypto for private key generation, storage and operation**

- The RACDCERT command with the GENCERT function will Generate a certificate and optionally a public-private key pair.
- Certificate information includes subject's name, certificate extensions, and the certificate validity period in the form of month, day, year and time of day.
- On the GENCERT, if SUBJECTSDN and *request-data-set-name* are not specified, the programmer name data from the ID() user (either specified or defaulted) is used as the common name (CN).
- When creating a public-private key pair, there are four options available for generating the key pair.
- If no key type is specified for key generation, the key is generated by software with RSA algorithm and stored in the RACF database. This is the default. If the key type keyword is DSA, the key is generated by software with the DSA algorithm and, again, stored in the RACF database. If the key type keyword is ICSF, the key is, again, generated by software with RSA algorithm, however, the key is stored in ICSF's PKDS. If the key type keyword is PCICC, the key is generated by the hardware with RSA algorithm and is stored in ICSF's PKDS. The keywords ICSF and PCICC utilizes the z/OS crypto facilities provided by ICSF for storage and/or key generation.
- PCICC is faster and more secure, but cannot be used with the CCFs that are pre z990.



IBM Systems™

IBM Education Assistant



RACDCERT GENCERT continued:

- **Certificate can be self signed or signed by another certificate's corresponding private key**
- **May create certificates for other servers, including those on non-z/OS platform**

© 2006 IBM Corporation

10

- Continuing with the GENCERT function ...

- Certificates can be self signed or signed by another certificate. For non self-signed certificates, the signing certificate needs to be specified using the SIGNWITH keyword. In order for the GENCERT function to successfully create a certificate from the certificate request, another certificate's corresponding private key is used to sign the new certificate. A public-private key pair is not generated for the non self-signed certificate. The certificate request contains the public key.
- For self-signed certificates, the dataset with a certificate request cannot be specified. A public-private key pair is generated for this certificate.
- Certificates that are created can be used for other servers, such as SSL and also exported to non-z/OS platforms.

RACDCERT GENCERT examples

- **Create a self signed certificate and public-private key pair using ICSF for a CA -- CERTAUTH represents the 'ID' of a CA, no 'SIGNWITH' keyword 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 'Our CA cert'**

- RACDCERT ID(WEBSRV) GENCERT SUBJECT(...) PCICC
SIGNWITH(CERTAUTH LABEL('Our CA cert'))...

•Let's look at some examples of how we can generate certificates.

•We need to create a certificate that we want to be able to be our certificate authority and sign certificates for us. We would also like to take advantage of storing our certificate authority's private key in an ICSF PKDS. To do so, we issue:

```
RACDCERT CERTAUTH GENCERT SUBJECT(CN('Our Certificate Authority')) ICSF  
WITHLABEL('Our CA cert')
```

•This creates a self signed certificate with the serial number of 0.

•Now we want to create a certificate for the ID WEBSRV and we want to have the public-private key pair generated by the crypto hardware and stored in ICSF's PKDS and have our own certificate authority that we just created sign it.

```
RACDCERT ID(WEBSRV) GENCERT SUBJECT(CN('Webserver') C('US'))  
WITHLABEL('WebServer1') PCICC  
SIGNWITH(CERTAUTH LABEL('Our CA cert'))
```

•The private key will be created by the Crypto hardware (z990 and above) and stored in the PKDS.

RACDCERT GENCERT examples:

- **Create a certificate based on a certificate request specified in the dataset 'CERTREQ.Acme' for User ID MYID . This generates the certificate only, no key pair is created.**

```
▪ RACDCERT ID(MYID) GENCERT ('CERTREQ.Acme')  
  SIGNWITH(CERTAUTH LABEL('Our CA cert'))...
```

•Continuing with our examples, If SIGNWITH is specified, it must refer to a certificate that has a private key associated with it. If no private key is associated with the certificate, an informational message is issued and processing stops. If *request-data-set-name* is specified on the GENCERT keyword, the SIGNWITH keyword is also required.

•Now let's suppose you have a certificate request that was given to you in an MVS dataset (CERTREQ.ACME) and you would like your own local certificate authority to sign this request making it into a certificate for the userID MYID. A certificate request already has a public key in it so no key pair is created. The syntax and format is as follows:

```
RACDCERT ID(MYID) GENCERT ('CERTREQ.ACME') WITHLABEL('AcmeCert')  
SIGNWITH(CERTAUTH  
  LABEL('Our CA cert'))
```

IBM Systems™ IBM Education Assistant IBM

RACDCERT GENREQ

- **GENerates a REQuest by copying information, including the public key, from a previously created certificate**
- **Syntax: RACDCERT GENREQ <label of cert used> <output dataset name>**
- **Usually issued after GENCERT a self signed place holder certificate, like**
 - `RACDCERT ID(outsrv) GENCERT SUBJECT(CN(...)) WITHLABEL('Outsrv Cert')`
- **When you issue the RACDCERT GENREQ:**
 - **The request is saved to a data set in the Base64 format which can be used in cut and paste**
 - **The created certificate request can be submitted (e.g. e-mail it, paste it into a web page, etc) to a CA for issuance**
 - **Note there is no key generation in GENREQ**
- ❑ **Example:**
 - **Generate a request based on an existing certificate named *My Self Signed Cert* in RACF owned by ID *OUTSRV* and put the request in the data set *CERTREQ.B64***
 - `RACDCERT ID(OUTSRV) GENREQ(LABEL('Outsrv Cert')) DSN(CERTREQ.mydataset)`

13

© 2006 IBM Corporation

•The last example, mentioned that we had a certificate request in a dataset that was sent to us. How can we create our own certificate request? We can do this by using the RACDCERT GENREQ command.

•In order to generate a request, you will need a certificate that has a private key associated with it.

•How we do that is to create a self-signed certificate with information that may be needed and used in the certificate such as the subject's name, extensions and so forth.

•First let's create a self-signed certificate for the user OUTSRV.

```
RACDCERT ID(OUTSRV) GENCERT SUBJECT(CN('Out Center Server'))
WITHLABEL('Outsrv Cert')
```

•Now we have a certificate with a public-private key pair. However, we need to have the certificate signed by an outside Certificate Authority. To do so, we need to create a request using 'Outsrv Cert' :

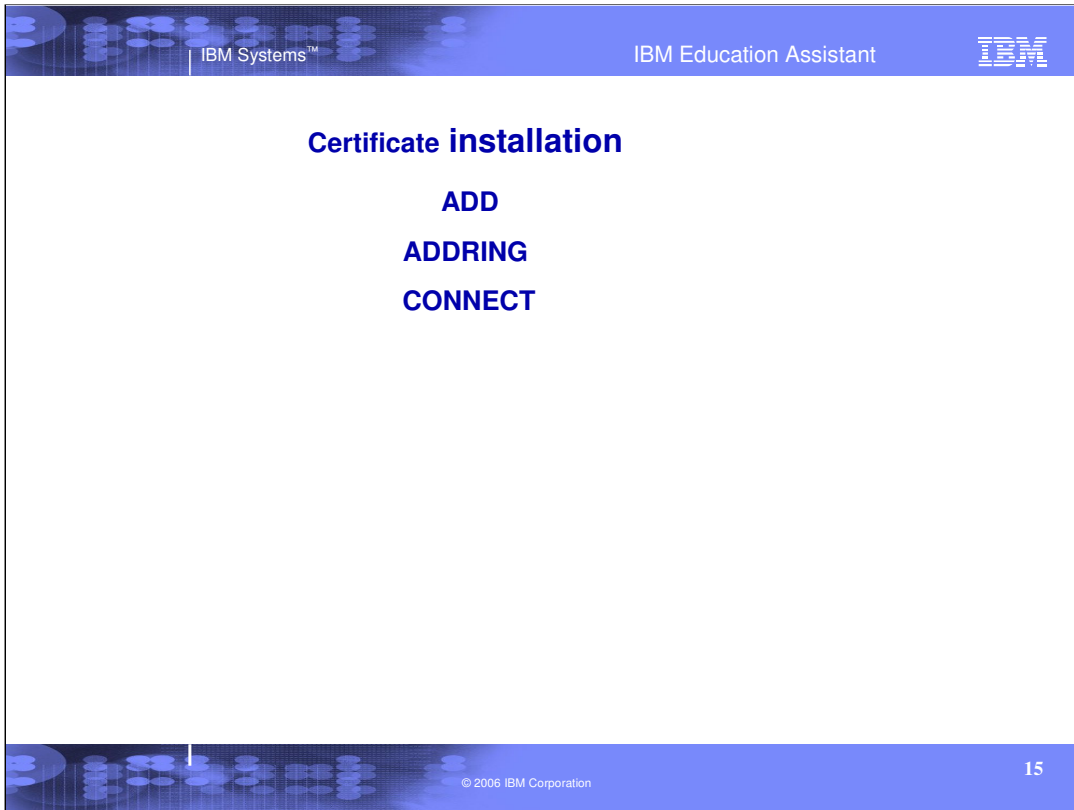
```
RACDCERT ID(OUTSRV) GENREQ (LABEL('Outsrv Cert')) DSN('certreq.mydataset')
```

•The request is now in a dataset that can be e-mailed, cut and pasted into a web page, etc. to be signed.

An example of a Base64 encoded certificate request

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBfzCB6QIBADAQMq4wDAYDVQQDEwV0ZXN0YTCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwgYkCgYEA3qy4qFTb97+kefbgMysxIXpaRQVwqT0I2XeDmIOWmXF6GkvK+i4k
7wr/pto+cGtqCzHsQrm6aRKiJF6pdizYkf4xew0DqdeVArOydr/4HESVN1JRqxJ/
jqY4IJ0uphnsbKKyfl7ny77u4M50YZBXRgq9VDAFpCaQbNW8xVkiUPECAwEAAAw
MC4GCSqGSIb3DQEJJDjEhMAAwHQYDVR0OBBYEFFX5QcyxyCL6q+NFFGjQpCnP9mB
MA0GCSqGSIb3DQEBAQUAA4GBAMyKoRZvGJAYVPummmMirJgMQ4KMcYrraI79rz7L
SWAq5/1Ppbkue9enn1xaS3eJZOQNXGaV4Rem3rGM740/PpQIF/qMN1pJZfOEzyL
rYxIaO6riEXM3Q2Y80m6C+X+V4k69eRC1LTvc8I515uz+EMCTd5x5PaGuzhXgjkE
Q5vt
-----END NEW CERTIFICATE REQUEST-----
```

- Creating a certificate for a server starts with the generation of a public/private key pair, and the creation of a certificate request also known as a PKCS#10 request which is done at the server end.
- Here's an example of one such request. It is base64 encoded (that is, it contains printable characters) so that it may be transported as text (pasted to a web page, sent as e-mail text, and so on.)



We have seen how we can create our own self-signed certificate, create a Certificate Authority certificate that will be used to sign other certificates and how to create a request that can be sent off to other Certificate Authorities to be signed. But what if we are not creating the certificates or requests ourselves but rather taking certificates that are already created and need to add them into the RACF database? And once the certificates we created and added are in the RACF database, how can they be known to the outside world? How can a server know how to find them?

We will now look at the adding of certificates plus the management of these certificates through key rings.

IBM Systems™ IBM Education Assistant IBM

RACDCERT ADD

- **Syntax: RACDCERT ADD <dataset contains cert package> <cert label to assign> <trust status> <password to open the package if needed> <store in ICSF flag>**
- **The certificate being installed can be a Personal, CERTAUTH or SITE certificate.**
- **Install a certificate to RACF with or without private key from a certificate package**
- **If the package contains a private key, a password is needed and you may choose if you want to store it in ICSF**

© 2006 IBM Corporation 16

- Using RACDCERT ADD is the way to import a certificate to RACF created by other Certificate Authorities from other platforms.
- You can import a single certificate (x509), a certificate chain (PKCS#7) or a certificate package with the private key (PKCS#12).
- A PKCS#7 certificate chain may have more than one certificate. The set consists of the hierarchy of Certificate Authorities. If the command issuer does not have the proper authority to add certificate authorities, but has the proper authority to add personal certificates, the first certificate (end entity) alone will be added.
- PKCS#12 certificate packages include a private key and a password and optionally other certificates. To add this type of certificate, the password must be used on the RACDCERT ADD command. Besides the personal certificate, there may also be Certificate Authority certificates in the PKCS#12 package. If the command issuer is authorized to add CERTAUTH certificates, the CA certificates will be sorted to determine the hierarchy chain. If the command issuer is not authorized to add CERTAUTH certificates, an informational message will be issued. In either case, processing will then continue with the first certificate in the package (the end-entity certificate).
- If you import a certificate package (PKCS#12), you can store the private key in ICSF's PKDS by specifying the ICSF keyword.

RACDCERT ADD Examples

- **Add a trusted CA certificate under ID *CERTAUTH*, specified in the dataset called *CA.CERT***
 - `RACDCERT CERTAUTH ADD (CA.CERT) TRUST...`
- **Add a trusted peer certificate under ID *SITE*, specified in the dataset called *SITE.CERT***
 - `RACDCERT SITE ADD (SITE.CERT) TRUST...`
- **Add a certificate package called *MY.CERT* that contains the private key under ID *myid***
 - `RACDCERT ID (myid) ADD (MY.CERT) PASSWORD ('secret')...`

Here we have some examples of certificates and how they are added.

- We talk about a ‘trusted CA certificate’. What is meant by trusted? Whether a certificate is not trusted or trusted depends on whether or not the certificate is valid and whether the private key has been compromised or not. That also includes the signer of the certificate. RACF in certain situations, will handle a certificate differently if it is marked ‘NOTRUST’. We will get into that during the key ring discussion. When a certificate is trusted, it can be used by RACF for its intended purpose (map to a user ID, or treat as a trusted certificate authority or trusted site). For a personal certificate, TRUST indicates that the certificate can be used to authenticate a user ID.
- To add a CERTAUTH certificate, one would issue the RACDCERT CERTAUTH ADD as seen above.
- To add a trusted peer certificate under ID of SITE, one would issue the RACDCERT SITE ADD as seen above.
- If you have a PKCS#12 certificate package that includes a private key, you will need to have the password that will enable you to decode and add the certificate. You would issue RACDCERT ADD with the keyword of PASSWORD(‘quoted string password’).

RACDCERT ADD examples:

➤ **Replace a previous self signed certificate under ID *OUTSRV*, with a new certificate specified in the dataset called *NEW.CERT***

▪ `RACDCERT ID(OUTSRV) ADD(NEW.CERT) TRUST`

❖ **Assuming the following steps were done:**

▪ `RACDCERT ID(OUTSRV) GENCERT(SUBJECT(CN(...)) WITHLABEL ('MYCERT'))`

▪ `RACDCERT ID(OUTSRV) GENREQ(LABEL('MYCERT')) DSN(CERTREQ.B64)`

▪ Send the dataset, *CERTREQ.B64*, containing the request, to your CA and get back a dataset, *NEW.CERT*, containing the certificate.

•As we have seen in the previous examples of RACDCERT GENCERT and RACDCERT GENREQ, we can create a self-signed certificate that has a public-private key pair, then create a certificate request out of that certificate. We can then, if needed, send that request to another certificate authority to be processed and signed. When we receive the new signed certificate, we can then do a RACDCERT ADD to replace the original self-signed certificate with the newly CA signed one. The newly signed certificate will still have the original public-private key pair associated with it.

RACDCERT ADDRING

- **Syntax: RACDCERT ADDRING <ring name>**
- **Create a key ring (for handshake process)**
- **Certificate must be placed in a key ring before it can be used by other middleware, e.g. SSL**
- **Example:**
- **Create a key ring called *SSLring* for ID *WEBSRV***
 - **RACDCERT ID(WEBSRV) ADDRING(SSLring)**

- Now let us look at how we can communicate our certificate and the status of it to others. A key ring is a RACF specific collection of keys and certificates used for one purpose or application.
- In order for a middleware application (for example, SSL) to use certificates, each of the communicating parties must have a keyring collection.
- To create a key ring, we do so with RACDCERT ADDRING. After the key ring is added, we must then populate the ring with certificates. We will demonstrate that shortly.
- Lower case characters are permitted, however, the ring name will become part of a RACF profile so only characters that are permitted in RACF profiles can be used as a ring name. A key ring name can be up to 237 characters in length. Since only user IDs can have key rings, neither CERTAUTH nor SITE can be specified with ADDRING.
- It is important to note that the ID() subcommand on RACDCERT ADDRING indicates the owner of the ring.

RACDCERT CONNECT

- **Syntax: RACDCERT <ringowner > CONNECT
(LABEL(' <cert label>') <Certowner>
RING(<ring name>) [<default flag>]
[USAGE (<usage flag>)])**
- **Place your own certificate in your own key ring**
 - **RACDCERT CONNECT (LABEL('MyCert')
RING(SSLring) USAGE(Personal))**

- Now that we have a ring, we need to add certificates to the ring. We do that with the RACDCERT CONNECT function.
- Looking at the above RACDCERT CONNECT command, let's assume that USER1 issued the command. We can see that we are adding the certificate MyCert to the ring sslRing with the usage of personal.
- Since we did not indicate anything differently, the owner of the ring is the command issuer, USER1. Additionally, we did not indicate the certificate owner so that also defaults to the command issuer. MyCert is owned by USER1.
- The USAGE keyword allows a certificate to be connected to a ring and used in a manner that differs from the certificate's original use. For example, a certificate that is a user certificate could be used as a certificate-authority certificate. The USAGE keyword allows the altering of the trust policy within the confines of a specific key ring.

RACDCERT CONNECT continued....

- **Example 1: You may link your certificates to your key ring**
 - RACDCERT CONNECT (LABEL('MyCert') RING(SSLring) USAGE(PERSONAL))
- **Example 2: You may link other's certificates to your key ring**
 - RACDCERT CONNECT (LABEL('User2 Cert') ID(USER2) RING(SSLring) USAGE(SITE))
- **Example 3: You may add your own certificate to a key ring owned by another user.**
 - RACDCERT ID(USER2) CONNECT (ID (USER1) LABEL('MyCert') RING(User2Ring))

- You may add your own certificate, certificates owned by others, SITE and CERTAUTH certificates to your key ring. In example 1, we have indicated that USER1 (the command issuer) is adding the certificate 'MyCert' which is owned by USER1 to USER1's ring SSLring with the usage of PERSONAL.
- In example 2, we have indicated that USER1 is adding the certificate 'User2 Cert' which is owned by USER2 to USER1's ring SSLring with the usage of SITE. Using the ID keyword within the parenthesis after the CONNECT indicates the identification of the certificate owner.
- You may also add certificates to rings that are not owned by the command issuer. Looking at the next example, Example 3, we can see that on the RACDCERT command, we indicated ID(USER2) before the CONNECT. This indicates the owner of the RING we are trying to connect the certificate to. In this case, we are connecting our own (USER1) certificate 'MyCert' to USER2's ring User2Ring.
- Since SITE and CERTAUTH can not own a ring, those keywords are ignored outside of the subfunctions.
- As you would imagine, you would need different RACF authorities to issue the above three examples.

IBM Systems™ IBM Education Assistant IBM

RACDCERT CONNECT continued ...

- **The TRUST status of the connected certificate in the ring indicates if the certificate can be used**
- **A certificate can be placed in more than one key ring**
- **You can pick one of the certificates to be the DEFAULT, some applications will depend on this attribute to indicate the signing certificate**

© 2006 IBM Corporation 22

- Continuing on with RACDCERT CONNECT, a certificate could be used as a certificate-authority certificate.
- We talked about the TRUST status of a certificate. This TRUST status indicates whether the certificate should be used. If the certificate becomes temporarily compromised, by altering it to NOTRUST, applications will not be able to use the certificate.
- A Certificate can be placed in many different rings. Additionally, each user may own multiple rings. Rings should be set up to indicate your Trust Policy.
- The DEFAULT keyword specifies that the certificate is the default certificate for the ring. Only one certificate within the key ring can be the default certificate. If a default certificate already exists, and another certificate gets added with the DEFAULT status, the original DEFAULT status is removed, and the newly added certificate becomes the DEFAULT certificate. If you want a certificate to be the default, DEFAULT must be explicitly specified.
- If you have a key ring with a default certificate and you want to remove the default status of the certificate without defining another certificate as the default certificate, CONNECT the certificate again without specifying the DEFAULT keyword.
- The certificate marked DEFAULT should have a private key associated with it.

RACDCERT CONNECT continued...

- **The usage of the certificate in the ring depends on the type of the certificate owner ID, unless overridden by the USAGE keyword:**
 - **Ordinary ID – implied usage is PERSONAL**
 - **Certificate Authority ID – implied usage is CERTAUTH**
 - **SITE ID – implied usage is SITE**
- **Certificates in the key ring may or may not have a private key associated with it.**

- The Usage keyword allows a certificate to be connected to a ring and used in a manner that differs from the certificate's original use.
- If the usage is PERSONAL, a certificate has private key associated with it, the purpose is to identify itself to another party.
- For usage CERTAUTH, a certificate may not have private key associated with it, the purpose is to validate another party's certificate signed by that CA certificate.
- For usage SITE, a certificate may not have a private key associated with it, the purpose is to short cut the validation process, for example, acceptance of an incoming certificate, even its signer certificate is absent, if that certificate is connected with usage SITE.
- Basically, there are two types of certificates in a key ring – certificates with a private key associated with it in RACF, and certificates that do not have a private key.
- In order to prove identity to other parties, Certificates need to have a private key available. To do this, the certificate must have a private key associated with it plus the certificate must be marked 'Personal' in the key ring. Data would be encrypted with the private key, sent to the other party and the other party would be able to be decrypt the data using the certificate's public key.
- Certificates without the availability of the private key can still be valuable in a key ring. These certificates can be used to validate a certificate that is sent by the other party. Typically, this would be a CA certificate, or, even possibly, a SITE certificate. The incoming certificate can be validated if it's CA certificate is in the ring.

RACDCERT CONNECT Examples

- **Connect WEBSRV's own certificate called *SSL cert* to WEBSRV's key ring called *SSLring* and make it the default certificate**
 - `RACDCERT ID(WEBSRV) CONNECT(LABEL('SSL cert'))
RING(SSLring) DEFAULT...`
- **Connect a certificate called *OtherCert* owned by ID *USERA* to WEBSRV's key ring called *SSLring***
 - `RACDCERT ID(WEBSRV) CONNECT(ID(USERA) LABEL('OtherCert'))
RING(SSLring)...`
- **Connect a CERTAUTH certificate called '*CA Cert*' to WEBSRV's key ring called *SSLring* with the usage to CERTAUTH.**
 - `RACDCERT ID(WEBSRV) CONNECT(CERTAUTH LABEL('CA Cert'))
RING(SSLring) USAGE(CERTAUTH)...`

➤ Let's construct a key ring for the purpose of preparing SSL communications. We previously created a ring called *SSLring* under the user ID *WEBSRV*. We need to connect *WEBSRV*'s 'SSL cert' to this key ring. The administrator would do so by issuing the `RACDCERT CONNECT` function. Looking at the first example, we are connecting the certificate 'SSL cert' to the ring *SSLring*, both owned by *WEBSRV*, as the `DEFAULT` certificate. Since the ring owner ID parameter was issued, the owner of the certificate defaults to the ring owner. If the user ID *WEBSRV* issued this command and had the proper authority, the ring owner ID would not be needed. The ID parameter for ring owner would default to the command issuer.

➤ Next example, another certificate owned by User ID *USERA* is connected to *WEBSRV*'s key ring. Notice that the first ID parameter indicates the owner of the key ring while the next ID parameter within the parenthesis indicates the owner of the certificate, in this case, *USERA*.

➤ Lastly, a `CERTAUTH` certificate is connected with the usage of `CERTAUTH`. Again, the owner of the certificate is within the parenthesis. The `USAGE(CERTAUTH)` keyword can be omitted in this example since the certificate owner is `CERTAUTH` and the `USAGE(CERTAUTH)` will be the default.



In order to administer the certificates and key rings within RACF, the subfunctions of the RACDCERT command are:

- LIST – List a certificate and it's associations in the database.
- ALTER – change the label or trust status.
- DELETE – remove the certificate from the database.
- REMOVE – take the certificate out of a key ring. The certificate still remains in the database.
- LISTRING – list the rings for a specific user with the certificate associations.
- DELRING – Delete a key ring.
- CHECKCERT – query a certificate in a dataset to see if it is in the RACF database.
- EXPORT – copy a certificate into a dataset.

RACDCERT LIST

❑ Syntax:

- **RACDCERT LIST <identifier of cert>**
- **Display certificate information for a userID**

❑ Example:

- **Display a certificate called *SSL Cert* for ID *WEBSRV***
 - **RACDCERT ID(WEBSRV) LIST(Label('SSL Cert'))**

•Certificates for a specific user, SITE or CERTAUTH can be listed. To list a specific certificate, use the identifier of the certificate. That identifier can be the label, or serial number and issuer's distinguished name combination. Obviously, using the label is more convenient. To see all the certificates for a specific user, SITE or CERTAUTH, leave off the identifier and all the certificates for that user will be displayed. The following is the information displayed: Owner of the certificate, Label, Trust status, Serial number, Subject's distinguished name, Issuer's distinguished name, the validity period, and if present, the type of private key, the Subject's AltNames, key usages, and key rings the certificates are connected to.

RACDCERT ALTER

❑ Syntax:

- **RACDCERT ALTER <identifier of cert> <new trust status> <new label>**
- **Change the TRUST/NOTRUST status or the label of a certificate**

❑ Example: Alter the status of a certificate called *SSL Cert* to NOTRUST for ID *WEBSRV*

```
▪ RACDCERT ID (WEBSRV) ALTER (Label ( 'SSL  
Cert ' ) ) NOTRUST
```

- To change the label of a certificate or to alter the status of a certificate, the function of ALTER is used. In this case, the identifier for a specific certificate is necessary.
- To disable the certificate for verification or identification purposes, alter the status to NOTRUST.
- RACF has pre-installed a number of well-known Certificate Authority certificates under CERTAUTH. Note that these CERTAUTH certificates do not have available private keys that exist in the system.
- You can connect those pre-installed CA certificates in your key ring. These CERTAUTH certificates are installed NOTRUST and would need to be altered to TRUST when used in the key ring.

RACDCERT DELETE

❑ Syntax:

- **RACDCERT DELETE <identifier of cert>**
- **Delete a certificate from RACF**

❑ Example:

- **Delete a certificate called *SSL Cert* for ID *WEBSRV***

```
▪ RACDCERT ID(WEBSRV) DELETE(LABEL('SSL  
Cert'))
```

•To remove a certificate from the RACF database, use the DELETE function. In order to delete a specific certificate, use the identifier (label or serial number/Issuer's distinguished name combination). If there is only one certificate for the user, the identifier can be left off. When a certificate is deleted, it is also removed from any of the key rings that it is connected to.

•To delete a certificate called 'SSL Cert' for ID WEBSRV, issue RACDCERT ID(WEBSRV) DELETE (LABEL('SSL Cert'))

RACDCERT REMOVE

❑ Syntax:

- **RACDCERT REMOVE <cert label> <ring name>**
- **Remove a certificate from a key ring**

❑ Examples:

- **Remove a certificate called *SSL Cert* for ID *WEBSRV* from its key ring called *SSLring***

- `RACDCERT ID(WEBSRV) REMOVE (LABEL ('SSL Cert') RING (SSLring))`

- **Remove a certificate called '*SignerCert*' owned by *CERTAUTH* from *WEBSRV*'s key ring called *SSLring***

- `RACDCERT ID(WEBSRV) REMOVE (CERTAUTH LABEL ('SignerCert') RING (SSLring))`

• In order to take a certificate out of the key ring, use the REMOVE function. After removing a certificate from a key ring, the certificate still exists in RACF.

• To remove a certificate called 'SSL Cert' for ID WEBSRV from the keyring SSLring owned by ID WEBSRV, issue RACDCERT ID(WEBSRV) REMOVE (LABEL ('SSL Cert') ring (SSLring)).

• To remove a certificated called '*SignerCert*' owned by *CERTAUTH* from *WEBSRV*'s key ring called *SSLring* issue RACDCERT ID(WEBSRV) REMOVE (CERTAUTH LABEL ('SignerCert') RING (SSLring)).

RACDCERT LISTRING

❑ Syntax:

- **RACDCERT LISTRING <ring name>**
- **Display the certificates connected to a key ring, the usage and the default flag**

❑ Examples:

- **Display a key ring called *SSLring* for ID *WEBSRV***
 - **RACDCERT ID(WEBSRV) LISTRING(SSLring)**
- **Display ALL the key rings for ID *WEBSRV***
 - **RACDCERT ID(WEBSRV) LISTRING(*)**

- The LISTRING function is used to see the key rings and the certificates connected to them. LISTRING will display all the certificates connected to the ring. For each certificate in the ring, the following information is displayed: The ring name, the owner of the certificate (ID(name), CERTAUTH, or SITE), the label assigned to the certificate, the DEFAULT status of the certificate within the ring, and the usage within the ring.
- To display a key ring called SSLring for ID WEBSRV issue: RACDCERT ID(WEBSRV) LISTRING (SSLring).
- To see all the rings associated with a user, RACDCERT LISTRING(*) can be used.
- Usually LISTRING is issued after CONNECT to verify the right connection has been made properly.

RACDCERT DELRING

❑ Syntax:

- **RACDCERT DELRING <ring name>**
- **Delete a key ring**

❑ Example:

- **Delete a key ring called *SSLring* from ID *WEBSRV***
 - **RACDCERT ID(WEBSRV) DELRING(SSLring)**

- To delete a specific key ring, use the DELRING function. After deleting a key ring, the certificates connected to it still exist in the RACF database.
- To delete a key ring called SSLring from ID WEBSRV, issue RACDCERT ID(WEBSRV) DELRING (SSLring).
- Note that only users can own key rings; CERTAUTH and SITE can not, therefore, those keywords will be ignored outside of the parenthesis.

RACDCERT CHECKCERT

❑ Syntax:

- **RACDCERT CHECKCERT <dataset contains the cert package> <password to open the package if needed>**
- **Display the certificate information that is contained in a dataset**
- **Indicate if the certificate has been installed**
- **Show the owner of the certificate if it is installed**

❑ Example:

- **Display the certificate information that is contained in a dataset called *CERT.FILE***
 - **RACDCERT CHECKCERT (CERT.FILE)**

•To see if a certificate in a dataset is installed in the RACF database, use the CHECKCERT function. If the certificate is installed in RACF, the information displayed will be the Owner of the certificate, Label, Trust status, Serial number, Subject's distinguished name, Issuer's distinguished name, the validity period, and if present, the type of private key, the Subject's AltNames, key usages, and key rings the certificates are connected to.

•Note if you do not have the proper authority to view the certificates, the CHECKCERT function will not display the RACF specific information, for example, owner of the certificate, label, trust status and if present, the type of private key, and key rings the certificates are connected to.

•To display the certificate information that is contained in a dataset called cert.file, issue RACDCERT CHECKCERT(CERT.FILE).

•

RACDCERT EXPORT

□ Syntax:

- **Syntax: RACDCERT EXPORT <cert label> <output dataset> <export format> <password to protect the cert package if needed>**
- **Write a certificate package stored in RACF to a dataset with or without the private key in a specified format**

□ Examples:

- **Write a certificate called *SSL Cert* for ID *WEBSRV*, without the private key, to a dataset called *SSL.CERT***
 - `RACDCERT ID(WEBSRV) EXPORT(LABEL('SSL Cert'))`
`DSN(SSL.CERT) FORMAT(CERTDER)`
- **Write the above certificate with the private key, to a dataset called *SSL.CERTKEY***
 - `RACDCERT ID(WEBSRV) EXPORT(LABEL('SSL Cert'))`
`DSN(SSL.CERTKEY) FORMAT(PKCS12DER) PASSWORD('secret')`

- The purpose of exporting a certificate package is to save it as a backup or to send it to some other applications to import. To do so, use the EXPORT function.
- The export formats can be in DER (binary) or B64 form, which is a Base64 encoding of the binary format.
- Supported formats are: X509 certificate (CERTDER, CERTB64), PKCS7 certificate chains (PKCS7DER, PKCS7B64), PKCS12 certificate package with private key (PKCS12DER, PKCS12B64).
- The PKCS7 keywords indicate to export a certificate and its CA chain. If the command issuer is authorized to export CERTAUTH certificates, PKCS#7 processing will attempt to package all the certificate authority certificates necessary to complete the chain. If a certificate in the chain cannot be found under CERTAUTH or is expired or the command issuer is not authorized to export CERTAUTH certificates, an informational message will be issued. Processing continues creating an incomplete PKCS#7 package. An incomplete PKCS#7 package can still be processed by RACF but may or may not be useful for OEM products.
- The PKCS#12 keywords indicate to export the certificate and the private key (which must exist and must not be an ICSF or PCICC key). The package produced by specifying one of the PKCS12 keywords is encrypted using the password specified according to the PKCS#12 standard. PKCS#12 processing will attempt to package any certificate-authority certificate necessary to complete the basing chain to the exported certificate. If a certificate in the chain cannot be found under CERTAUTH, an informational message will be issued. Processing continues and an incomplete PKCS#12 package is created that can still be processed by RACF but may or may not be useful for OEM products.
- Different applications may accept different package formats, you need to know what the application needs so that you can export the certificate in the required format.

ADVANCED RACDCERT FUNCTIONS

- RACDCERT REKEY
- RACDCERT ROLLOVER
- RACDCERT MAP
- RACDCERT ALTMAP
- RACDCERT LISTMAP
- RACDCERT DELMAP

- There are other more advanced functions available with the RACDCERT command. REKEY and ROLLOVER functions are used for renewing certificate and keys and the MAP, ALTMAP, LISTMAP and DELMAP functions are used for certificate mapping, associating user IDs to special criteria.
- This brings us to the end of this presentation. See Digital Certificates and RACF Part III, for more information and examples. Part III will also go through some actual examples and ideas on how to renew certificates, log in a system without a user ID and password, how to share a certificate's associated private key in a key ring and how to use the mapping functions of RACDCERT.

References

- ❑ **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)
- ❑ **Cryptographic Services**
 - PKI Services Guide and Reference (SA22-7693)
 - OCSF Service Provider Developer's Guide and Reference (SC24-5900)
 - ICSF Administrator's Guide (SA22-7521)
 - System SSL Programming (SC24-5901)
- ❑ **RACF web site:**
 - <http://www.ibm.com/servers/eserver/zseries/zos/racf>
- ❑ **PKI Services web site:**
 - <http://www.ibm.com/servers/eserver/zseries/zos/pki>
- ❑ **PKI Services Red Book:**
 - <http://www.redbooks.ibm.com/abstracts/sq246968.html>
- ❑ **Other Sources:**
 - PKIX - <http://www.ietf.org/html.charters/pkix-charter.html>

Here is a list of references to learn more about Digital Certificates, Security products such as RACF, and other Cryptographic services available with z/OS.