CA Top Secret® for z/OS

Cookbook

r15

Seventh Edition

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the “Documentation”), is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This documentation set references the following CA products:

- CA ACF2™ for z/OS (CA ACF2)
- CA Common Services for z/OS (CA Common Services)
- CA Distributed Security Integration Server for z/OS (CA DSI Server)
- CA LDAP Server for z/OS (CA LDAP Server)
- CA Top Secret® for z/OS (CA Top Secret)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following changes have been made since the last release of this documentation

- Allow UNIX Users to Increase the Limit on the Number of Mutexes (see page 31)—Added this section.

The following changes were made in the previous release of this documentation:

- PKCS 7 and PKCS 12 Certificates (see page 76)—Added this section.
- Define a System Default UID and GID (see page 15)—Clarified that OMVSGRP and OMVSUSR are not supported in z/OS 2.1 and above, in which case you can use UNIQUSR and MODLUSER instead.
- Allow UNIX Users to Change File Ownership (see page 28)—Added syntax for permitting the user with the appropriate access level; defined each access level and the actions that each level allows; added an example.
- Add a Certificate to an ACID (see page 77)—Updated the DCDSN description with information about PKCS 7 and PKCS 12 certificate packages (including information about certificate authority chains); updated the PKCSPASS description; expanded HITRUST|TRUST|NOTRUST description; expanded LABLPKDS description.
- List Digital Certificate Information (see page 94)—Added CHAIN to syntax; added information about GENREQ, which indicates that the CERTDATA record has been the target of a GENREQ command.
- Remove a Certificate from a User (see page 103)—Mentioned the following restriction: If the GENREQ command has been issued against a certificate (when creating a certificate based on an original certificate), the original certificate cannot be removed unless you specify the FORCE operand or the new certificate has replaced the original certificate.
- Determine Certificate Associations (see page 109)—Added CHAIN keyword, which displays information for each certificate in the chain of the input data set and displays summary Information as applicable.
- Obtain Security Information for a File within OpenEdition (see page 177)—Updated this section to indicate how the system displays displays information when the UID is 0.

The following changes were made in the previous release of this documentation:

- CPF Limitations (see page 71)—Added details about which certificate-related commands (and keywords) are sent through CPF and which commands are not sent through CPF.

Contents

Chapter 5: Digital Certificates 69

Chapter 1: z/OS UNIX System Services

This section contains the following topics:

UNIX Security

z/OS UNIX System Services (USS) allows UNIX applications to run on a z/OS mainframe. UNIX security is based on users and groups having a unique binary identifier, a User ID (UID) or a Group ID (GID).

CA Top Secret supports the following services in USS:

- Callable services
- Digital Certificates
- Hierarchical File System (HFS)
- Home and Path definitions
- UID and GID definitions
- USS Auditing
- USS Security Trace Facility
- USS MVS Shell Setup Utility (ISHELL)

Controlling Access to USS

When a user attempts to enter the USS shell, CA Top Secret verifies that the user:

- Is a USS user before the system initializes the shell
- Associated with a program attempting to access a USS resource is a USS user before allowing access to the requested resource

Defining an ACID as a USS User

To define an ACID as a USS user:

- Define the user to CA Top Secret
- Assign a USS Group
- Assign a USS UID to the user
- Assign the user to a default group

Define USS Users

USS recognizes ACIDs by their assigned UID. The OMVS segment of a user ACID defines the UID, the user's home directory, and the initial program that the user runs. The initial program is generally the shell program that the user invokes.

Note: The MSCA ACID *cannot* be used to sign on to USS.

To define a USS user, enter the following command:

```
TSS ADD(acid) UID(user_id)
              [HOME(/u/pathname)]
              [OMVSPGM(/bin/sh)]
```

acid

Identifies the ACID that you are defining as a USS user.

UID(*user_id*)

Specifies a unique numeric ID for maintaining individual accountability and control in USS. All ACIDs require a UID. You cannot assign a UID value that is already assigned to another ACID, unless the value is 0. Specifying 0 indicates that the user is a superuser that passes all USS security checks and can access all UNIX files.

Important! To eliminate unauthorized access risks, any ACID that is assigned UID(0) should also be assigned a non-expiring password. You can issue command TSS REPL(*acid*) PASS(*xxxx*,0), where *acid* identifies the ACID receiving the password and *xxxx* specifies the password.

Other than the required started task ACIDs, all ACIDs should have non-zero UIDs and be permitted the necessary authorities in CA Top Secret resource class IBMFAC plus file permissions. IBMFAC is equivalent to the IBM FACILITY class.

Range: 0 to 2,147,483,647

HOME

(Optional) Specifies the path name of the initial directory that is used when a user enters the OMVS command or enters the ISPF shell. You can use uppercase and lowercase characters. If you do not define HOME, USS sets the initial directory for the user to the root directory.

Range: 1 to 1024 characters

OMVSPGM

(Optional) Specifies the user's USS shell program. This program is the first program started when the user specifies the OMVS command or a USS batch job is started (using the BPXBATCH program). You can specify uppercase and lowercase characters. If you do not define OMVSPGM, USS gives control to the default shell program.

Range: 1 to 1024 characters

Example: Define a USS User

This example defines user OMVSU2 as a regular user:

```
TSS ADD(OMVSU2) UID(199)
                HOME(/u/omvsu2)
                OMVSPGM(/bin/sh)
```

Example: Define a Superuser

This example defines user OMVSUSR as a superuser:

```
TSS ADD(OMVSUSR) UID(0)
```

HOME and OMVSPGM are not defined, so USS assigns defaults for these fields.

Define USS Groups

CA Top Secret requires a unique GID for each group ACID.

The GROUP type ACID:

- Defines USS groups to CA Top Secret
- Contains the OMVS segment which consists of one field: the GID keyword

GID is a numeric field that accepts values from zero to 2,147,483,647.

You can assign up to 256 groups to a user.

Important! At least one OMVS group ACID must exist prior to implementation.

Example: creating and assigning a group

This example creates an OMVS GROUP ACID for a group called OMVSGRP and assign it a GID of 1.

```
TSS CREATE(OMVSGRP) TYPE(GROUP)
                    NAME('OMVSGROUP')
                    DEPT(OMVSDEPT)

TSS ADD(OMVSGRP) GID(1)
```

Assign Users to Groups

To assign a user's default group, set the DFLTGRP field in the user's ACID.

Example: assigning a user to a group

This example assigns ACID OMVSU2 to group OMVSGRP and then assigns the group as its default:

```
TSS ADD(OMVSU2) GROUP(OMVSGRP)

TSS ADD(OMVSU2) DFLTGRP(OMVSGRP)
```

Define a System Default UID and GID

CA Top Secret provides support for default UID and GID through the OMVSUSR and OMVSGRP control options. Evaluate your security policy to determine whether all users should be given their own UIDs and GIDs. Overuse of the default feature limits your ability to audit access permissions under USS.

Note: OMVSGRP and OMVSUSR are not supported in z/OS 2.1 and above. For more information about UNIQUSER and MODLUSER, see the *CA Top Secret Control Options Guide*.

Use the following process to define a default UID and GID:

1. Define an ACID with a valid OMVS segment (for example, a UID, a HOME, and an OMVSPGM). Specifying this ACID on the OMVSUSR option makes the ACID the default to use for any users without such a segment. You make the specification dynamically or through the TSS PARMFILE, and an example is as follows:

   ```
   TSS MODIFY(OMVSUSR(acid_name))
   ```

2. Extract the OMVS segment from a group by performing one of the following actions:

   - Use control option OMVSGRP:

     ```
     TSS MODIFY(OMVSGRP(group_acid))
     ```

   - Add the default group to the OMVSUSR ACID:

     ```
     TSS ADD('acid_name') DFLTGRP('group_acid')
     ```

   Both methods define a TYPE GROUP ACID.

3. (Optional) Add the NOOMVSDF attribute to the user ACID to prevent a user with no UID or group from using the default values:

   ```
   TSS ADD(acid) NOOMVSDF
   ```

If you define the BPX.DEFAULT.USER profile, all users will have access to z/OS UNIX. To limit access, define an OMVS segment with no UID. This prevents unauthorized users from using a UNIX service. If users must have anonymous access (for FTP or other socket use) without using the shell, define the initial program for the default user as /bin/echo.

AutoUID and AutoGID

The ADD and REPLACE commands auto-assignment of UID/GID numbers means that:

- Users can define a default range
- Users can restrict the search for an open number by entering a specific range
- If no number is assigned and no range is given, AutoUID/GID restricts the search by checking through the defined default range
- If no range is given and there is no default range, AutoUID/GID starts at 1 and searches until an available UID/GID is found
- Zeros are excluded in the range search

Examples: Auto-assignment of UID and GID

```
TSS ADD|REP(acid|Group Acid) UID|GID(?) RANGE(<low>,<high>)
or
TSS ADD|REP(acid|Group Acid) UID|GID(?)
or
TSS ADD|REP(acid|Group Acid) UID|GID(?) RANGE(300,400)
```

Refresh UID and GID Tables

During initial startup, CA Top Secret builds in-storage tables of UIDs and GIDs (and their related ACIDs). CA Top Secret automatically updates the tables in the following situations:

- A TSS ADD, REMOVE, or REPLACE command specifies the UID or GID keyword.
- A TSS DELETE command deletes an acid with a UID or a GID.
- The IBM getGMAP or getUMAP callable service assigns a UID or GID to a user or group acid.

If an error occurs during an automatic update, you can manually refresh the tables. A manual refresh rebuilds the tables with the current information from the security file.

To refresh the tables, enter the following command:

```
TSS MODIFY(OMVSTABS)
```

A message indicates that the operation was successful. The table refresh is complete.

More information:

UNIX Security (see page 11)

List UIDs and GIDs

To list of all UIDs enter:

```
TSS WHOOWNS UID(*)
```

To list of all GIDs enter:

```
TSS WHOOWNS GID(*)
```

List Who Has UID(0)

To list of all users with UID(0) enter:

```
TSS WHOHAS UID(0)
```

BPX Facility Resource Classes

The BPX facility resources classes are:

BPX.SUPERUSER

Allows non-superusers to gain superuser authority. (Control over UNIX command *su*).

BPX.DAEMON

Allows daemon programs to validate user password and then change the identity of a spawned address space (control over *setuid ( )* and *seteuid ( )*).

BPX.SERVER

Allows daemon programs to customize the security environment of a thread.

BPX.SMF

Restricts access for C/C++ applications to generate SMF records without requiring APF authorization.

BPX.DEBUG

Allows users to use dbx to debug programs that run as APF-authorized or with BPX.SERVER authority.

BPX.WLMSERVER

Allows users to use WLM server functions.

BPX.STOR.SWAP

Allows users to make address spaces non-swappable.

BPX.FILEATTR.APF

Allows users to turn on the APF-authorized attribute for an HFS file.

BPX.FILEATTR.PROGCTL

Allows users to turn on the program controlled attribute for an HFS file.

BPX.JOBNAME

Controls which users are allowed to set their own job names by using the BPX.JOBNAME environment variable or the inheritance structure on spawn. Users with READ or higher permissions to this resource can define their own job names.

For information on the BPX facility resources classes, see the *z/OS UNIX Systems Services Planning Guide*.

Examples: BPX facility resource classes

This example establishes BPX.SUPERUSER ownership:

```
TSS ADD(dept) IBMFAC(BPX.)
```

This example grants access to specific resources:

```
TSS PER(acid) IBMFAC(BPX.SMF)
```

This example restricts access for C/C++ applications to generate SMF records without requiring APF authorization.

```
TSS PER(acid) IBMFAC(BPX.SMF) ACC(READ)
```

Password Assignment for UID(0) Acids

Unauthorized access can occur with ACIDs defined with NOPW and UID(0).

To eliminate this security concern, add passwords to all ACIDs assigned with UID(0):

```
TSS REPL(acid) PASS(xxxx,0)
```

Password Prompts

Several created started task ACID definitions specify a password. Started task ACIDs with passwords cause a password prompt at console startup. Prompting is optional and can be turned off using control option setting OPTIONS(4).

The OPTIONS control option must be set via the CA Top Secret parameter file. It cannot be set with a MODIFY command.

Define the OMVS Started Task ACID for Using CA Top Secret in USS

To use CA Top Secret in a USS environment, you must assign an ACID to USS to use for the OMVS started task.

Follow these steps:

1. Create group ACIDS to which to attach the started task ACID:

```
TSS CREATE(OMVSGRP)  TYPE(GROUP)
                     NAME('OMVS GROUP')
                     DEPT(OMVSDEPT)
TSS CREATE(TTY)              TYPE(GROUP)
                             NAME('REQ OMVS TTY GROUP')
                             DEPT(OMVSDEPT)
```

2. Assign GIDs to the group ACIDs:

```
TSS ADD(OMVSGRP) GID(1)
TSS ADD(TTY) GID(2)
```

   The product assigns GID 1 to the OMVSGRP group and assigns GID 2 to the TTY group.

3. Create the ACID to use for the OMVS started task:

```
TSS CREATE(OMVSKERN) TYPE(USER)
                     NAME('OMVS STC ACID')
                     PASS(password,0)
                     DEPT(dept)
                     FACILITY(STC)
```

4. Define the OMVS started task to the STC record:

```
TSS ADD(STC) PROCNAME(OMVS)
             ACID(OMVSKERN)
```

5. Issue the following command to assign superuser status, assign a default group, and assign the OMVSGRP and TTY groups to the OMVS started task ACID:

```
TSS ADD(OMVSKERN) UID(0)
                      DFLTGRP(OMVSGRP)

                      GROUP(OMVSGRP)
                      GROUP(TTY)
```

6. Define a BPXROOT ACID:

```
TSS CREATE(BPXROOT) TYPE(USER)
                    NAME('BPXROOT ACID')
                    PASS(password,0)
                    DEPT(OMVSDEPT)
TSS ADD(BPXROOT) GROUP(OMVSGRP)
                 DFLTGRP(OMVSGRP)
                 UID(0)
```

Defining this ACID ensures that the system can function properly if the SUPERUSER parameter is not defined in SYS1.PARMLIB member BPXPRM*xx*. If SUPERUSER has no assigned value, the default is BPXROOT.

Important! Your defined BPXROOT ACID must be defined with UID(0) and must *not* have BPX.DAEMON authorization.

You have defined the OMVS started task ACID.

TSO ISHELL Support

The CA Top Secret TSSOPMAT file contains a replacement member for the IBM REXX exec BPXWIRAC. Copy TSSOPMAT(BPXWIRAC) to a partitioned data set compatible with your TSO SYSEXEC data sets. Assure that this file is concatenated ahead of the IBM supplied SYSEXEC data sets. Failure to do this results in a S0C1 ABEND when entering the ISHELL.

Define Other OMVS Started Task ACIDs

To define ACIDs for other OMVS started tasks, enter:

```
TSS CRE(INETD) TYPE(USER)
               NAME('OMVS INETD STC')
               DEPT(dept)
               FAC(STC)
               PASSWORD(password,0)

TSS ADD(INETD) UID(0)
               GROUP(OMVSGRP)
               DFLTGRP(OMVSGRP)
               HOME(/)
               OMVSPGM(/bin/sh)

TSS CRE(RMFGAT) TYPE(USER)
                NAME('OMVS RMFGAT')
                DEPT(dept)
                FAC(STC)
                PASSWORD(password,0)

TSS ADD(RMFGAT) UID(0)
                GROUP(OMVSGRP)
                DFLTGRP(OMVSGRP)
                HOME(/)
                OMVSPGM(/bin/sh)

TSS ADD(STC) PROCNAME(INETD)
             ACID(INETD)

TSS ADD(STC) PROCNAME(RMFGAT)
             ACID(RMFGAT)

TSS ADD(STC) PROCNAME(BPXOINIT)
             ACID(OMVSKERN)

TSS ADD(STC) PROCNAME(BPXAS)
             ACID(OMVSKERN)
```

Create the Superuser Administrator ACID

The USS and utilities installation process must have superuser authority to create directories in the Hierarchical File System (HFS). Assigning superuser authority to an ACID does not give the user any authority within CA Top Secret

Important! Use caution when assigning ACIDs superuser authority. A superuser can access any UNIX file in the file system.

To create a superuser administrator ACID

1. Enter:

   ```
   TSS ADD(SYSPROG1) UID(0)
   ```

   SYSPROG1 is defined as a superuser.

2. Enter:

   ```
   TSS ADD(SYSPROG1) GROUP(OMVSGRP)
                     DFLTGRP(OMVSGRP)
   ```

   SYSPROG1 is defined as a member of a group. SYSPROG1 can sign on and be validated as a member of group OMVSGRP. The ACIDs of group OMVSGRP are a special subset of users who perform system-related tasks.

Superuser Granularity

Superuser Granularity support lets you avoid giving users superuser authority via UID(0) by allowing non-superuser users access to new resources in the UNIXPRIV class. If a user does not have a UID=0, but they do have access to one of the new resources, access is allowed.

Activating CA Top Secret SAF HFS security does not override the superuser granularity support if there is an equivalent SAF HFS security resource for the UNIXPRIV resource.

If there is no SAF HFS resource, the UNIXPRIV resource is checked instead.

Resources and Access Relationships

The following table shows the new resources and the access allowed by the resource.

Resource Name	Access Given	Functions Affected
SUPERUSER.FILESYS.FILE (READ access or higher)	Allows a user to read any HFS file and read or search any HFS directory	Open*(for read, opendir(), readlink(), stat(), realpath(0)
SUPERUSER.FILESYS.FILE (UPDATE access or higher)	Allows a user to write to any existing HFS file.	Open() for write

Resource Name	Access Given	Functions Affected
SUPERUSER.FILESYS.FILE (CONTROL Access)	Allows a user to write to any HFS directory.	Link(), mkdir(), rename(), mdir(), syslink(), unlink()
SUPERUSER.FILESYS.CHOWN	Allows a user to change ownership of any file.	Chown()
SUPERUSER.FILESYS.MOUNT	Allows a user to issue mount, unmount, quiesce, and unquiesce requests. change ownership of any file.	Mount(), unmount(), quiesce(), unquiesce()
SUPERUSER.FILESYS.PFSCTL	Allows a user to call pfsctl()	Pfsctl()
SUPERUSER.FILESYS.VREGISTER	Allows a user to issue vregister() to register as a vfs file server	Vregister()
SUPERUSER.IPC.RMID	Allows a user to do ipcrm calls to clean up leftover IPC mechanisms	Ipcrm command user of IPC_RMID for msgct(), semctl(), shmctl()
SUPERUSER.PROCESS.GETPSENT	Allows users to see all processes	Getpsent()—ps command
SUPERUSER.PROCESS.KILL	Allows user to send signals to any process	Kill()
SUPERUSER.PROCESS.PTRACE	Allows users to use dbx to trace any process	Dbx
SUPERUSER.SETPRIORITY	Allows a user to increase his priority.	Setpriority(), nice()

Examples: superuser granularity

Before you can give a user access to a SUPERUSER resource in the UNIXPRIV class, enter:

```
TSS ADD(dept) UNIXPRIV(SUPERUSE)
```

This example gives USER01 authority to read to any HFS file:

```
TSS PERMIT(USER01) UNIXPRIV(SUPERUSER.FILESYS.FILE)
                   ACCESS(READ)
```

This example gives USER01 authority to change ownership of a file:

```
TSS PERMIT(USER01) UNIXPRIV(SUPERUSER.FILESYS.CHOWN)
                   ACCESS(READ)
```

USS User Limits

Use the parmlib settings defined in BPXPRMxx to control the amount of resources consumed by individual z/OS UNIX users.

The new resources and the access allowed by the resource are:

ASSIZE

Specifies the maximum address space region size allowed per process created via rlogin or telnet.

Range: 10,485,760 to 2,147,483,647

Member: Maxassize

MMAPAREA

Specifies the maximum amount of dataspace storage (pages) that can be allocated for memory mapping of HFS files.

Range: 1 to 16,777,216

Member: Maxmmaparea

OEFILEP

Specifies the maximum number of files that a single process can have active or open concurrently.

Range: 3 to 65,535

Member: Maxfileproc

PROCUSER

Specifies the maximum number of processes a user can have open at the same time.

Range: 3 to 32,767

Member: Maxprocuser.

OECPUTM

Specifies the maximum time (seconds) a process is allowed to use.

Range: 7 to 2,147,483,647

Member: Maxcputime

THREADS

Specifies the maximum number of pthread_created threads, including those running, queued, and exited but not detached, that a single process can have concurrently active.

Range: 0 to 100,000

Member: Maxthreads

Example: USS user limits

This example limits USER01 to a maximum of 200 open processes at the same time|

```
TSS ADD(USER01) PROCUSER(200)
```

This example remove the above PROCUSER authorization:

```
TSS REMOVE(USER01) PROCUSER
```

z/OS ServerPac Upgrade

Prior to restoring the HFS, ensure that the proper authority is given to the user ID that submits the dialog jobs**.**

For z/OS 1.7 and above, the user ID must be a superuser with UID(0) or have access to the BPX.SUPERUSER resource in the FACILITY.

For previous releases of z/OS, the user ID must be a superuser (UID=0). Having access to the BPX.SUPERUSER facility class is not sufficient because the PAX utility that unloads the serverpac HFS does not yet use the BPX.SUPERUSER facility class to establish superuser identification.

To authorize the ACID to run the PAX utility

1. Enter:

   ```
   TSS ADD(SYSPROG) UID(0)
   ```

   ACID SYSPROG1 is defined as a superuser.

2. Enter:

   ```
   TSS ADD(SYSPROG) GROUP(OMVSGRP)
                    DFLTGRP(OMVSGRP)
   ```

   SYSPROG1 is defined as a member of a group.

3. Enter:

   ```
   TSS PERMIT(SYSPROG) IBMFAC(BPX.FILEATTR.APF)
                       ACC(READ)

   TSS PERMIT(SYSPROG) IBMFAC(BPX.FILEATTR.PROGCTL)
                       ACC(READ)
   ```

   or

   ```
   TSS PERMIT(SYSPROG) IBMFAC(BPX.FILEATTR.)
                       ACC(READ)
   ```

   The IBMFACs are authorized.

Logging USS Security Calls

To implement audit within USS at the file or directory level use:

```
CHAUDIT
```

This command specifies audit options for individual files or directories.

Once audit is set for a file or directory using the CHAUDIT command, SMF records are written for the file or directory designated activity. This can include access and changes to permission bit settings.

Tracing USS (OMVS)

You can use SECTRACE to trace SAF requests made by OMVS. The only allowable value for the DEST= parameter of the TYPE=OMVS SECTRACE command is DEST=SYSLOG.

Note: Only use the OMVS SECTRACE when instructed to by CA Top Secret Technical Support due to the large volume of trace entries possible in the OMVS environment. It is usually easier to debug an OMVS problem using the TSSOERPT report, because it shows more information than the trace. All of the OMVS services write SMF records when the service returns with a non-zero return code.

To start SECTRACE for OMVS, enter:

```
ST SET,ID=xxxx,TYPE=OMVS,FUNC=ALL,END
```

xxxx can be:

ALL

Traces all OMVS services.

CHANGE

Traces R_chown, R-chaudit, and R_cmod.

CHECK

Traces ck_access, ck_priv, ck_process_owner, ck_file_owner, R_ptrace, ck_IPC_access, ck_owner_two_files, R_IPC_ctl, and R_dceauth.

GET

Traces getUMAP, getGMAP, R_getgroups, R_getgroupsbyname, get_uid_gid_supgrps, R_dceinfo, R_dcekey, R_dceuid, and R_usermap.

INIT

Traces initACEE, initUSP, deleteUSP, and R_fork.

MAKE

Traces makeFSP, makeISP, and make_root_FSP.

MISC

Traces audit, query_file_security_options, and query_system_security_options.

SET

Traces R_umask, R_setegid, R_seteuid, R_setgid, R_setuid, R_exec, clear_setid, and R_admin.

To stop the SECTRACE for OMVS, enter:

```
ST DISABLE,ID=xxxx,END
```

To restart a disabled trace, enter:

```
ST ENABLE,ID=xxxx,END
```

xxxx is the identifier assigned to the SECTRACE

USS Reporting

CA Top Secret logs security events under USS to SMF using the standard CA Top Secret SMF record. Log records are written for any security event that denies the ACID access to a USS facility. Use the records to determine the UID and GID of the ACID involved in the attempted access.

The TSSOERPT batch utility program processes security-related activity recorded in SMF data sets. For information on TSSOERPT, see the *Report and Tracking Guide*.

Manage UNIX with UNIXPRIV Class Profiles

Resource names in the UNIXPRIV class are associated with UNIX privileges. Define profiles in the UNIXPRIV class to:

- Grant superuser privileges to specified users who do not have superuser authority. This minimizes the number of superuser authority assignments and reduces your security risk.
- Use CA Top Secret authorization to grant UNIX privileges.

Example: authorizing superuser privileges

This example gives user01 superuser privileges to the UNIX chown command.

1. Enter the command:

   ```
   TSS ADD(UNIXDEPT) UNIXPRIV(SUPERUSER.FILESYS.CHOWN)
   ```

   A profile is defined in the UNIXPRIV class to protect the resource called SUPERUSER.FILESYS.CHOWN.

2. Enter the command:

   ```
   TSS PERMIT(user01) UNIXPRIV(SUPERUSER.FILESYS.CHOWN ACCESS(READ)
   ```

   READ access is the only access allowed. User01 is allowed to issue the chown command to change ownership of any file.

Allow UNIX Users to Change File Ownership

CA Top Secret enforces the rules for the _POSIX_CHOWN_RESTRICTED constant. However, you can disable the constant, allowing users to change file ownership (depending on the access level that you permit).

Follow these steps:

1. Define the CHOWN.UNRESTRICTED resource:

   ```
   TSS ADD(dept) UNIXPRIV(CHOWN.UN)
   ```

   dept

   Specifies the department ACID to which you are assigning ownership of the resource.

   Having this resource defined means that _POSIX_CHOWN_RESTRICTED is not in effect.

2. Permit the user to the resource (with the appropriate access level):

   ```
   TSS PERMIT(acid) UNIXPRIV(CHOWN.UNRESTRICTED)
   ACCESS(READ|UPDATE)
   ```

   acid

   Specifies the ACID to which you are providing the permit and access.

   READ

   Lets users change ownership of files they own to any non-zero UID value or to the GID of a group to which the user is not connected.

   UPDATE

   Lets users change ownership of files they own to UID 0.

Example: Permit a UNIX User to Transfer File Ownership to Any UID or GID on a System

This example allows a UNIX user to transfer ownership of files they own to any UID (except 0) or GID on the system:

```
TSS PERMIT(USERX) UNIXPRIV(CHOWN.UNRESTRICTED)
ACCESS(READ)
```

Specify the Group Owner for New UNIX Files

When a new UNIX file is created:

- The owning UID is initialized from the effective UID of the creating process
- The owning GID is copied from the parent directory

The POSIX standard allows the owning GID to be taken from either the parent directory or the effective GID of the creating process.

To specify that the group owner of a new UNIX file comes from the effective GID of the creating process, define the FILE.GROUPOWNER.SETGID resource in the UNIXPRIV class.

To set up the FILE.GROUPOWNER.SETGID resource enter:

```
TSS ADD(UNIXDEPT) UNIXPRIV(FILE.GROUPOWNER.SETGID)
```

To specify that the group owner of new objects within the file system comes from the parent directory, turn on the set-gid bit of its root directory when a new file system is mounted.

ACL Administration

To create and administer an ACL for a file, you must have one of the following:

- File ownership
- Superuser authority by having UID(0)
- READ access to SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class

To use ACLs in access decisions, activate the HFSACL control option. You can define and display ACLs with HFSACL off, however they are not used for authorization checking.

If you have defined default ACLs on directories, the ACLs are inherited by new objects when HFSACL is set off but they are not used for authorization checking.

To activate the HFSACL control option, enter:

```
F TSS,HFSACL(ON)
```

Access Control to File System Resources

Users with the RSTDACC attribute cannot access protected resources they are not specifically authorized to access.

The RSTDACC attribute has no effect when a user accesses a UNIX file system resource. The file's “other” permission bits can allow access to users who are not explicitly authorized.

To prevent restricted users from accessing file system resources

1. Enter the command:

   ```
   TSS ADD(UNIXDEPT) UNIXPRIV(RESTRICTED.FILESYS.ACCESS)
   ```

   The RESTRICTED.FILESYS.ACCESS resource is defined in the UNIXPRIV class.

2. (Optional) Enter the command:

   ```
   setfacl -m user:thabo:rwx MyFile
   ```

   The specified restricted users are permitted to access to files by adding them or one of their groups to the file's ACL.

   Authorization changes made using the setfacl command take effect immediately.

3. (Optional) Enter the command:

   ```
   TSS PERMIT(userid) UNIXPRIV(RESTRICTED.FILESYS.ACCESS) ACCESS(READ)
   ```

   The specified restricted users are granted access based on the file's “other” bits.

Override SUPERUSER.FILESYS.FILE Authority

A user denied access through the ACL can access a file system resource if they have sufficient authority to the SUPERUSER.FILESYS.FILE resource in the UNIXPRIV class.

To prevent users from using SUPERUSER.FILESYS.FILE authority to access file system resources:

1. Enter the command:

   ```
   TSS ADD(UNIXDEPT) UNIXPRIV(SUPERUSER.FILESYS.ACLOVERRIDE)
   ```

   The SUPERUSER.FILESYS.ACLOVERRIDE resource is defined in the UNIXPRIV class.

2. (Optional) Enter the command:

   ```
   TSS PERMIT(userid) UNIXPRIV(SUPERUSER.FILESYS.ACLOVERRIDE) ACCESS(READ)
   ```

   The specified users or groups are granted exceptions to allow them to gain access based on their SUPERUSER.FILESYS.FILE authority. Permit users or profiles to the resource with the same level of access they require for the SUPERUSER.FILESYS.FILE resource.

Allow UNIX Users to Increase the Limit on the Number of Mutexes

z/OS Unix has increased the limit on the number of mutexes or condition variables that a user can define. The increased limits are only for users who are authorized to those new limits. A new UNIXPRIV profile, SUPERUSER.SHMMCV.LIMIT has been defined to control authorization to this ability.

To allow UNIX users to increase the limit on the number of mutexes or condition variables, enter:

```
TSS ADD(UNIXDEPT) UNIXPRIV(SUPERUSER.SHMMCV.LIMIT)
TSS PER(USER1) UNIXPRIV(SUPERUSER.SHMMCV.LIMIT)
```

Chapter 2: Using TCP/IP

This section contains the following topics:

TCP/IP Security

The process to establish a proper security connection is:

- Define TCP/IP to CA Top Secret by adding a facility definition for TCP/IP to the Facility Matrix Table.

  For example, rename a USER*xx* entry:

  ```
  FAC(USER11=NAME=TCP)
  FAC(TCP=PGM=xxx)
  FAC(TCP=NOTSOC,RES,NOIJU,AUTHINIT)
  ```

  Depending on the release of TCP/IP used, the program name (*xxx*) is:

  TCP/IP 3.1 PGM=MVP

  TCP/IP 3.2 PGM=EZA

  TCP/IP 3.4 PGM=EZB

  Set all other FACILITY parameters according to your site-specific needs.

  Note: If the RES attribute is omitted from the FACILITY definition, no user or profile data set rules are loaded into the TCP address space.

- Create a Region ACID.

  For example:

```
TSS CRE(TCP) NAME('TCPIP/FTP
             REGION ACID')
             FAC(BATCH,STC)
             PASS(password,0)
             DEPT(DEPT)
             MASTFAC(TCP)
             NOVOLCHK
             NODSNCHK
             NOLCFCHK
             NORESCHK
             NOSUBCHK

TSS ADD(TCP) UID(0)
             GROUP(OMVSGRP)
             DFLTGRP(OMVSGRP)
             HOME(/)
             OMVSPGM (/bin/sh)
```

  The Region ACID must have:

  - The NODSNCHK attribute or a permit for DSN(*****) ACC(ALL).
  - The NOSUBCHK attribute. If this attribute is omitted, DRC157 errors on submit of batch jobs from the TCP address space occur.

  To properly secure job submission, the JES facility must be in FAIL MODE.

- Define the TCP procedure to the CA Top Secret STC Record.

  For example:

```
TSS ADD(STC) PROCNAME(TCPPROC)
             ACID(TCP)
```

TCP/IP SERVAUTH Class

The SERVAUTH resource class protects TCP/IP resources from unauthorized access. The functions protected are:

Stack Access

Controls which users can get access to the TCP/IP stack.

Resource name: EZB.STACKACCESS.sysname.tcpipid

Net Access

Controls which users can access individual networks.

Resource name: EZB.NETACCESS.sysname.tcpipid.netname

Port Access

Controls which users can use TCP and UDP ports.

Resource name: EZB.PORTACCESS.sysname.tcpipid.portname

TN3270

Controls which users can use the secured ports.

Resource name: EZB.TN3270.sysname.tcpipid.PORT*nnnn*

sysname

Specifies the name of the system

tcpipid

Specifies the name of the TCP/IP started task

netname

Specifies the network name in PROFILE.TCPIP

portname

Specifies the port name in PROFILE.TCPIP

nnnn

Specifies the port number with leading zero's

HFS Access

Controls which users can use FTP to access HFS.

VMCF and TNF Subsystems (CA Top Secret started before JES)

If using restartable VMCF and TNF subsystems, add the VMCF, TNF, and EZAZSSI task entries to the CA Top Secret STC table:

```
TSS ADD(STC) PROCNAME(VMCF)
             ACID(acid)

TSS ADD(STC) PROCNAME(TNF)
             ACID(acid)
```

The ACIDs associated with these STCs require access to facility STC and data set hlq SEZATCP:

```
TSS ADD(dept) DSNC(SEZATCP.)

TSS PER(acid) DSN(SEZATCP.)
              ALL(READ)
```

IP Address Protection

Securing an IP address using CA Top Secret requires that the TCP/IP product installed pass the IP address packet. IBM's TCP/IP product does pass the IP address packet.

Important! IP address protection is not available if your TCP/IP product does not pass the IP address packet.

The IP packet passed is generated from the user's originating IP address. The IP packets often have no resemblance to standard LU names. Each node of the IP address is translated into a character representation of the hex value of the node. For example, the IP address 141.202.201.56 appears as terminal 8DCAC938.

IP Source Restriction

To restrict a user to enter the system only through a given IP address, use source restriction.

Example: IP source restriction

This example restricts user01 to 141.202.201.56:

TSS ADD(user01) SOURCE(8DCAC938)

Terminal Restriction

To protect an IP address or range from use, use terminal restriction.

Examples: terminal restriction

This example restricts use of all IP addresses starting 141.202 for all users:

```
TSS ADD(dept) TERMINAL(8DCA)
```

This example permits userid2 to use IP addresses starting 141.202:

```
TSS PERMIT(userid2) TERMINAL(8DCA)
```

This example permits userid3 to use IP addresses starting 141.202.201:

```
TSS PERMIT(userid3) TERMINAL(8DCAC9)
```

Using FTP

FTP is a feature of TCP/IP that allows users to transfer files to and from the mainframe. In addition, remote users can submit jobs to MVS. Users are required to identify themselves when using FTP.

FTP runs as an MVS or USS application. Security configuration is similar for both.

Secure FTP

FTP runs as its own started task which needs to be associated with a region ACID and the TCP/IP facility.

The command used to create this ACID should look like:

```
TSS CRE(FTP) NAME('FTP SERVER ACID')
             FAC(BATCH,STC)
             PASS(password,0)
             DEPT(DEPT)
             MASTFAC(TCP)
             NOVOLCHK
             NODSNCHK
             NOLCFCHK
             NORESCHK
             NOSUBCHK
```

Note: The use of the bypass attributes such as NODSNCHK and NOSUBCHK, are entered for simplicity. You can choose to omit them and explicitly permit the ACID to all resources it will access.

To define the FTP procedure to the CA Top Secret STC record enter:

```
TSS ADD(STC) PROCNAME(FTPSERVE)
             ACID(FTP)
```

Secure FTP for USS

OE/FTP is an OMVS application that executes under USS to facilitate the file transfer of HFS files throughout a TCP/IP network. OE/FTP typically executes under a started-task FTPD whereas FTP typically executes under a started-task named FTPSERVE.

OE/FTP includes an optional message-log daemon (Syslog-D) that logs past and present message traffic related to OE/FTP. Without this logging task, no ongoing log of OE/FTP activity occurs.

To replace the IBM requirements when installing OE/FTP with CA Top Secret:

- The O/E FTP started task (daemon) typically runs under a user ID of FTPD. The exception occurs when the task is automatically started by OMVS, in which case it inherits the identity of the OMVS kernel, typically OMVS or OMVSKERN.

  If running as an FTPD, the following example shows the administration needed to properly define the ACID:

```
TSS CRE(FTPD) TYPE(USER)
              NAME('OE/FTP STC ID')
              DEPT(anydept)
              FAC(STC)
              PASSWORD(password,0)
              MASTFAC(TCP)

TSS ADD(FTPD) UID(0)
              GROUP(OMVSGRP)
              DFLTGRP(OMVSGRP)

TSS ADD(STC)  PROCname(FTPD)
              ACID(FTPD)
```

  If running under the OMVS kernel ID, no additional setup is necessary.

- The FTP server ACID requires the ability to perform work on behalf of users logging on to FTP, and at times switch its identity to that of a user. Accordingly, it requires superuser authority. This can be done by adding UID(0) to the ACID or giving the ACID a non-zero UID and permitting it access to superuser authority as follows:

```
TSS PERMIT(FTPD) IBMFAC(BPX.SUPERUSER)
                 ACCESS(READ)
```

  IBM recommends that you increase your level of security by protecting daemon authority by owning the resource IBMFAC(BPX.DAEM). To explicitly permit daemon authority to the server, even if it is running under UID(0), enter the following command:

```
TSS PERMIT(FTPD) IBMFAC(BPX.DAEMON)
                 ACCESS(READ)
```

  IBM also documents the requirement for the FTPD user ID to have access to the FACILITY class resource of BPX.POE; therefore, the following permission may be required:

```
TSS PERMIT(FTPD) IBMFAC(BPX.POE) ACCESS(READ)
```

- The TSS TCP facility controls which end users are able to access OE/FTP. Access to the TCP facility must be given to user ACIDs that access OE/FTP.

Considerations for Securing FTP

Users accessing FTP must log on to the service before using it. This requires them to supply their userids and passwords and for their userids to have access to the TCPIP facility.

FTP also supports anonymous logons. In the FTPDATA configuration file, the parameter ANONYMOUS controls whether this feature can be used. The parameter can be specified in one of three ways:

- The parameter is specified without a following userid, and an FTP user specifies anonymous at logon time, RACINIT are issued for the ACID ANONYMOU. If this ACID is defined to CA Top Secret and has access to the TCP/IP facility, the user is allowed to log on and run under the authority of the “ANONYMOU” ACID.

  ```
  ANONYMOUS
  ```

- The parameter is specified with a userid. The user must provide the correct password for this userid.

  ```
  ANONYMOUS userid
  ```

- The parameter is specified with both USERID and PASSWORD, these values are used to sign on the user.

  ```
  ANONYMOUS userid/password
  ```

ANONYMOUS logon should be carefully considered, and if used, is a candidate for auditing.

Terminal Source Restriction

FTP logons specify a terminal ID when logging on a user. This terminal ID supplied is generated from the user's originating IP address. Terminal IDs often have no resemblance to standard LU names. Each node of the IP address is translated into a character representation of the hex value of the node.

Example: terminal source restriction

IP address 141.202.201.56 appears as terminal 8DCAC938. To administer source protection in CA Top Secret, enter:

```
TSS ADD(acid) SOURCE(8DCAC938)
```

TELNET

TELNET allows users terminal access to a system over a TCP/IP network. TELNET runs under both native MVS and USS.

If running under native MVS:

- Users are forced to log on to TELNET itself if the TELNET configuration statements in the TCP/IP profile data set do not specify a DFLTAPPL. If configured this way users require the TCP/IP facility.
- Users logon to a session manager such as CA-TPX™ if DFLTAPPL is specified. The session manager then controls access through its security interface.

Secure TELNET for USS

When using TELNET under OMVS, RLOGIN runs under its own ACID until the user successfully signs on. The ID of this ACID is specified in the configuration file etc/inetd/conf. TELNET is delivered specifying an ID of OMVSKERN and must be defined with both superuser and daemon authority.

Example: defining an ACID with dual authority

In this example an ACID is defined with both superuser and daemon authority:

```
TSS CRE(OMVSKERN) TYPE(USER)
                  NAME('OMVS ACID')
                  PASS(password,0)
                  DEPT(dept)

TSS ADD(OMVSKERN) UID(0)
                  GROUP(OMVSGRP)
                  DFLTGRP(OMVSGRP)
                  HOME(/)
                  OMVSPGM(/bin/sh)
```

If you are using OMVSKERN, it is likely that this ID was defined as part of your OMVS installation.

If you are securing daemon authority, the TELNET server ID must have the permission:

```
TSS PER(OMVSKERN) IBMFAC(BPX.DAEMON)
```

Chapter 3: WebSphere

This section contains the following topics:

Network Considerations

WebSphere for z/OS supports access to resources by clients and servers in a distributed network.

In a distributed network:

- Authorize servers to the base operating system services in z/OS. These services include CA Top Secret security, database management, and transaction management.
- For the servers, distinguish between:
  - Control regions. These run authorized system code, so they are trusted.
  - Server regions. These run application code and are given access to resources, carefully consider the authorizations you give server regions.
- Distinguish between the level of authority given to run-time servers compared to your own application servers. For example, the System Management server needs the authority to start other servers, while your own application servers do not need this authority.
- Authorize clients (users) to servers and objects within servers. The characteristics of each client requires special consideration:
  - Is the client on the local system or is it remote? The security of the network becomes a consideration for remote clients.
  - Will you allow unidentified (unauthenticated) clients to access the system? Some resources on your system can be intended for public access, while others must be protected. In order to access protected resources, clients must establish their identities and have authorization to use those resources.
  - What kind of objects will the client access? Enterprise beans and CORBA objects have differing authorization mechanisms.

Resource Protection

To protect resources, identifying who accesses the resources is critical. The security system requires client (user) identification, also known as authentication. In a distributed network supported by WebSphere for z/OS, clients can access resources from:

- Within the same system as a server
- Within the same sysplex as the server
- Remote z/OS systems
- Heterogeneous systems, such as WebSphere on distributed platforms, CICS, or other CORBA-compliant systems.

Clients can request a service that requires a server to forward the request to another server. In such cases the system must handle delegation (the availability of the client identity for use by intermediate servers and target servers).

Message Protection

In a distributed network, ensure that:

- Messages are confidential and have not been tampered
- Clients are who they claim to be
- Network identities are mapped to z/OS identities correctly

These issues are addressed by the following support in WebSphere for z/OS:

- The use of SSL and digital certificates
- Kerberos
- Distributed Computing Environment (DCE)

WebSphere Security

Network security is not required for your initial installation and customization of WebSphere for z/OS. This information is provided to introduce you to WebSphere for z/OS security and allow you to make early planning decisions about system security.

Authorization Checking

Each control region, server region, and client must have its own MVS user ID. When a request flows from a client to the server or from a server to a server, WebSphere for z/OS passes the user identity (client or server) with the request. Each request is performed on behalf of the user identity and the system checks to see if the user identity has the authority to make such a request. The following table shows the control/authorization relationships:

Control	Authorization
Access control lists in LDAP	Controlled access to WebSphere for z/OS naming and interface repository data
CBIND class	Access to a server
DATASET class	Access to data sets
DCEUUIDS and IBMFAC classes	Mapping DCE credentials to Top Secret user Ids
DSNR class	Access to DB2
EJBROLE class	Access to methods in enterprise beans
IBMFAC (IRR.DIGTCERT.GENCERT) & (IRR.DIGTCERT.LIST) & (IRR.DIGTCERT.LISTRING)	SSL key rings, certificates and mappings
IBMFAC class (IMSXCF.OTMACI)	Access to OTMA for IMS access
IBMFAC Class (IRR.RUSERMAP)	Kerberos credentials
File permissions	Access to HFS files
GRANTs (DB2)	DB2 access to plans and database
LOGSTRM class	Access to log streams
OPERCMDS class	Start and stop servers by Daemon
PTKTDATA class	Passticket enabling in the Sysplex (This relates to the session keys in the NDT in Top Secret)
SERVER class	Access to control region by a server region
SOMDOBJS class	Access to methods in CORBA objects
STC	Associate procname and userid in the STC table
SURROGAT class (*.DFHEXCI)	Access to EXCI for CICS access

Server Authorization Checking

To control access to WebSphere for z/OS resources, give:

- Greater authority to the control regions. These contain WebSphere for z/OS system code, are trusted, deal with multiple users, have greater authorization and run APF-authorized.
- Less authority to the server regions. These contain application code, are not trusted (other than having authorization to get work and to attach to data stores), and should run unauthorized.

WebSphere for z/OS Runtime Servers

For WebSphere for z/OS, run-time servers give less authority to the daemon and naming server, and greater authority to the system management server as the following table shows:

Run-Time Server	Region	Required Authorities
Daemon Server	Control	STC Control, access WLM services, access to DNS, OPERCMDS access to START, STOP, CANCEL, FORCE & MODIFY other services
Naming Server	Control	STC Control STARTED class, access to WLM services
Naming Server	Server	STC control, READ auth to the SERVER class, DBADM for the LDAP Database
Sys. MGMT. Server	Control	STC Control
Sys. Mgmt. Server	Server	Started class, Read auth. to the SERVER class, OPERCMDS access to START, STOP, CANCEL FORCE and MODIFY Other services
Interface Repository Server	Control	STARTED class
Interface Repository Server	Server	STC Control, READ Auth. to the SERVER Class, DBADM for the LDAP database

Assigning Authorities

When assigning authorities to WebSphere for z/OS run-time server control and server regions protect the:

- RRS log streams
- WebSphere for z/OS environment files, especially if they contain passwords

User Identification, Authentication, and Network Security

LDAP uses access control lists to control client access to naming services. Usually, you set up a general ANYBODY user identity with read access to the LDAP name space, allowing any client to access naming services.

CBIND Class

Use the CBIND class to restrict a client's ability to access servers. WebSphere for z/OS uses two types of resources in the CBIND class:

CB.BIND.*server_name*

This form controls whether a local or remote client can access servers.

CB.*server_name*

This form controls whether a client can use objects in a server.

When you add a new server, authorize all systems management user IDs (for example, CBADMIN) to have read access to the CB.*server_name* and CB.BIND.*server_name* resources.

Example: CBIND authority

In this example CBADMIN needs read authority to the CB.BBOASR1 and CB.BIND.BBOASR1 servers:

```
TSS ADD(deptacid) CBIND(CB.)

TSS PERMIT(CBADMIN) CBIND(CB.BBOASR1)
                    ACCESS(READ)

TSS PERMIT(CBADMIN) CBIND(CB.BIND.BBOASR1)
                    ACCESS(READ)
```

EJBROLE Class

Use the EJBROLE (or GEJBROLE) class in CA Top Secret to control a client's access to enterprise beans.

To protect an application using EJB roles:

- Define a profile name using the EJBROLE (or GEJBROLE) class. For example:

  `TSS ADD(`*`deptartment`*`) EJBROLE(`*`role_name`*`)`

  Department

  Specifies a department already defined in the CA Top Secret database.

  role_name

  Matches the security role attribute specified in the jar file or for the application. Role names can be in mixed case but cannot contain blanks.

  Range: Uo to 245 charactrers

- Create membership in the role by permitting CA Top Secret userID's or profiles permission to the defined EJBROLE resource. For example:

  `TSS PERMIT(acid) EJBROLE(`*`role_name)`*

- The application assembler must assign method permissions to the bean or method using the Application Assembly Tool. Define the roles relevant to the application. The role names must match the resource names defined to CA Top Secret. Once defined, the role can be assigned to access an application (as a method permission). After the application assembly is complete, re-install the application using the Administration application.

SOMDOBJS Class

Use the SOMDOBJS class in CA Top Secret to control a client's access to CORBA objects. Resource names in SOMDOBJS have the form:

```
server_name.home.method
```

server_name

Specifies the server name.

Size: 8 characters or less

home

Specifies the home name.

Size: 192 characters or less

method

Specifies the method name.

Size: Up to the remainder of 244 minus the sum of the server and home name lengths.

Example: If the server name is eight characters, and the home name is 128 characters, the method name can be 244 - 8 -128 = 108.

If a method is protected by SOMDOBJS and a client program is using the method to:

- Give the client UPDATE authorization for the method.
- Read an attribute of an object, give the client READ authorization for the method.

All names are folded into uppercase characters, regardless of how you enter them. There is no difference between MY_server.MY_home.MY_method and MY_SERVER.MY_HOME.MY_METHOD.

In addition to the SOMDOBJS definitions, specify method-level access checking through the WebSphere for z/OS Administration application. Check the box for method-level access checking when you define your application's container.

Resource Managers

Resource managers such as DB2, IMS, and CICS have implemented their own resource controls, which control the ability of clients to access resources.

When resource controls are used by DB2, use the DSNR CA Top Secret class or issue the relevant DB2 GRANT statements.

Access to OTMA for IMS access is through the IBMFAC Class (IMSXCF.OTMACI). Access to EXCI for CICS is through the SURROGAT class (*.DFHEXCI).

Control access to data sets through the DATASET class and HFS files through file permissions or the HFSSEC class.

Identification

For identification, each control region and server region start procedure must have its own user ID and be defined in the STC record. Control regions are trusted, server regions are not. Because you should give differing resource authorizations to each, give different user IDs to control regions and server regions.

Additional user IDs are required for installation. Definitions for these user IDs are provided in the CA Top Secret sample. For information, see the customized instructions produced when you run the customization dialog.

Authentication

An operator starts a server by using the START command and the control region start procedure. Authentication of the start procedure's user ID is made because an operator started the start procedure—no password is required. To restrict an operator's ability to start servers, use the STC record in CA Top Secret.

WASADM

The CA Top Secret CLIST, WASADM is supplied in the SAMPJCL file. This CLIST runs and executes the CA Top Secret commands required to create the environment for the Websphere implementation.

Security Auditing

Security auditing is handled by the security product. WebSphere for z/OS uses the System Authorization Facility (SAF), which provides an auditing mechanism consistent with other functions in z/OS.

Chapter 4: Servers

This section contains the following topics:

CA LDAP Server for z/OS

CA Top Secret permits secured access to user information through standard LDAP protocols. For example, an LDAP session can query, delete, add, and modify information including user-defined fields stored within the CA Top Secret ACID record. CA clients are able to take advantage of these LDAP capabilities using the CA supplied LDAP-compliant directory server for the z/OS platform.

The CA LDAP Server for z/OS provides:

- Integration with CA Common Services for z/OS
- Access control for directory information
- Strong LDAP server authentication
- Interoperability with both CA and third party LDAP clients
- A high-performance repository

InfoPrint Server for z/OS

The print server is InfoPrint Server for z/OS. Control of the resources defined under the print server requires the definition of the groups:

AOPOPER

The operator group with authority to start and stop the print interface.

AOPADMIN

Provides authority to administer the printer inventory and controls.

If the separation of authority is not necessary, one group name can be defined for both functions.

Define the Security Environment

To define the security environment for CA Top Secret, enter:

```
TSS CRE(AOPADMIN) TYPE(GROUP)
                  NAME('PRINT SERVER')
                  DEPT(dept)

TSS ADD(AOPADMIN) GID(6)

TSS ADD(admin acid) GROUP(AOPADMIN)

TSS CRE(AOPER) TYPE(GROUP)
               NAME('PRINT SERVER')
               DEPT(dept)

TSS ADD(AOPER) GID(7)

TSS ADD(acid) GROUP(AOPER)

TSS ADD(JDCSYS) IBMFAC(AOPADMIN)

TSS PERMIT(dept acid) IBMFAC(AOPADMIN)
                      ACCESS(ALL)
```

Lotus Domino Go Webserver

The Lotus Domino Go Webserver allows the MVS mainframe to act as an Internet web server. This is installed and managed as a USS application. By default, the Domino Go Webserver materials are installed within the OMVS environment in /usr/lpp/internet.

Domino Go Webserver installation includes several steps using the RACF security administrator.

Domino Go Webserver Installation

The examples in the following steps reflect default procnames, typical group names, and typical GID value. These commands ensure that a valid OMVS UID and GID exist for each of the started tasks that access OMVS.

The process to install Domino Go Webserver is:

- Ensure that previously defined USS and TCP/IP requirements are completed.
- Create a TSS FACILITY for the web server. Use this facility to add each user ACID that is allowed to log on to the web server. Tailor the following command and then add it to the existing CA Top Secret startup control options:

  ```
  TSS MODIFY FAC(USERx=NAME=IMWEB)
  ```

- The Domino Go Webserver requires an ACID for the web server started task and for a web administrator. Both of these ACIDs must be connected to an OMVS Group ID for the web server. The following commands accomplish this using the IBM default values.

  The IMWEBSRV web server started task is referred to by IBM as the web server daemon. Also, changing the ID of the web administrator is recommended; however, this change must be coordinated with updates to the web server configuration file.

  ```
  TSS CRE(IMWEB) TYPE(GROUP)
                 NAME('WEBSERVER GROUP')
                 DEPT(anydept)

  TSS ADD(IMWEB)  GID(205)

  TSS CRE(WEBADM) TYPE(USER)
                  NAME('WEB ADMINISTRATOR')
                  DEPT(anydept)
                  FAC(IMWEB)
                  PASSWORD(password,0)

  TSS ADD(WEBADM) UID(206)
                  GROUP(IMWEB)
                  DFLTGRP(IMWEB)
                  HOME(/usr/lpp/internet)
                  OMVSPGM(/bin/sh)

  TSS CRE(WEBSRV) TYPE(USER)
                  NAME('WEBSERVER
                  DAEMON/STC')
                  DEPT(dept)
                  FAC(STC,IMWEB)
                  PASSWORD(password,0)
  ```

```
TSS ADD(WEBSRV) UID(0) GROUP(IMWEB)
                DFLTGRP(IMWEB)
                HOME(/usr/lpp/internet)
                OMVSPGM(/bin/sh)
                MASTFAC(IMWEB)
TSS ADD(STC) PROCNAME(IMWEBSRV)
             ACID(WEBSRV)
```

- Three other user ACIDs, each having their own connected group, are required unless “surrogate user” support is disabled. This feature permits users to access the web server without requiring a signon. Disable this feature for security reasons.

  To disable this feature, change the “Userid” option to “%%CLIENT%%” within the web server configuration file. See IBM documentation. If not disabled, the following commands will create ACIDs and groups for surrogate support following IBM examples:

```
TSS CRE(EXTERNAL) TYPE(GROUP)
                  NAME('WEB GROUP')
                  DEPT(dept)

TSS ADD(EXTERNAL) GID(999)

TSS CRE(EMPLOYEE) TYPE(GROUP)
                  NAME('WEB GROUP')
                  DEPT(dept)

TSS ADD(EMPLOYEE) GID(500)

TSS CRE(SPECIAL) TYPE(GROUP)
                 NAME('WEB GROUP')
                 DEPT(dept)

TSS ADD(SPECIAL) GID(255)

TSS CRE(PUBLIC) TYPE(USER)
                NAME('WEB SURROGATE ID')
                DEPT(dept)
                FAC(IMWEB)
                PASSWORD(NOPW,0)

TSS ADD(PUBLIC) UID(998)
                GROUP(EXTERNAL)
                DFLTGRP(EXTERNAL)
                HOME(/)
                OMVSPGM(/bin/sh)
```

```
TSS CRE(INTERNAL) TYPE(USER)
                  NAME('WEB SURROGATE ID')
                  DEPT(dept)
                  FAC(IMWEB)
                  PASSWORD(NOPW,0)

TSS ADD(INTERNAL) UID(537)
                  GROUP(EMPLOYEE)
                  DFLTGRP(EMPLOYEE)
                  HOME(/)
                  OMVSPGM(/bin/sh)

TSS CRE(PRIVATE) TYPE(USER)
                 NAME('WEB SURROGATE ID')
                 DEPT(dept)
                 FAC(IMWEB)
                  PASSWORD(NOPW,0)

TSS ADD(PRIVATE) UID(416)
                 GROUP(SPECIAL)
                 DFLTGRP(SPECIAL)
                 HOME(/)
                 OMVSPGM(/bin/sh)
```

- The ACID for the web server started task (that is, Daemon) requires access to the following IBMFAC and SURROGAT resources. Note that the final three permits are not needed if surrogate support is disabled:

```
TSS ADD(dept) IBMFAC(BPX.)

TSS PERMIT(WEBSRV) IBMFAC(BPX.DAEMON)
                   ACCESS(READ)
TSS PERMIT(WEBSRV) IBMFAC(BPX.SERVER)
                   ACCESS(UPDATE)

TSS ADD(dept) SURROGAT(BPX.)

TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.WEBADM)
                   ACCESS(READ)

TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.PUBLIC)
                   ACCESS(READ)

TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.PRIVATE)
                   ACCESS(READ)

TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.INTERNAL)
                   ACCESS(READ)
```

- IBM documentation includes:
  - One install step where “RACF program control” is discussed. In the first part of this step, all users are given read access to three load libraries related to the web server. This part translates to:

    ```
    TSS ADD(dept) DSNAME(CEE.)

    TSS ADD(dept) DSNAME(IMW.)

    TSS ADD(dept) DSNAME(SYS1.)

    TSS PERMIT(ALL) DSNAME(CEE.V1R5M0.SCEERUN)
                    ACCESS(READ)

    TSS PERMIT(ALL) DSNAME(IMW.V1R1M0.IMWMOD1)
                    ACCESS(READ)

    TSS PERMIT(ALL) DSNAME(SYS1.LINKLIB)
                    ACCESS(READ)
    ```

    This step also describes several (RDEFINE and SETROPTS) commands needed to exempt the above libraries from RACF “PADS” checking. These commands are not applicable to CA Top Secret and can be skipped.

    (To RACF, these commands mark all programs in these libraries as “NOPADCHK”. This means that any program-restricted data set access should not have to list any of the programs from these libraries. In other words, this marks all programs from these libraries as being trusted and therefore exempt from any program accessed data set / PADS checks. These commands are not applicable to CA Top Secret.)
  - Install steps, which discuss permission bits are related to OMVS file security itself and are unrelated to RACF. Follow these steps as described.

Lotus Notes Server

The Lotus Notes Server (email) can run on a z/OS environment. The external security interface requires a facility and a DOMINO console interface (identified in IBM as DOMCON). This interface facilitates sending commands from z/OS to stop, start, and manage Lotus Notes Server running under USS.

To set up Lotus Notes server

To create an ACID for the server started task and a group ACID for the Lotus Notes Server using IBM default values, enter:

```
TSS CREATE(LOTUSGRP) TYPE(GROUP)
                     NAME(LOTUSGROUP)
                     DEPT(OMVSDEPT)

TSS ADD(LOTUSGRP) GID(6789)

TSS CREATE(DOMCON) TYPE(USER)
                   NAME('LOTUS STC ACID')
                   PASS(password,0)
                   DEPT(OMVSDEPT)
                   FACILITY(STC)

TSS ADD(DOMCON) GROUP(LOTUSGRP)
                DFLTGRP(LOTUSGRP)

TSS ADD(STC) PROCNAME(?????)
             ACID(DOMCON)
```

(Repeat this command for all LOTUS PROCs. There can be multiple procs associated with this address space beginning with “DOMIN”.)

```
TSS ADD(DOMCON) UID(0)
                HOME(/u/domcon)
                OMVSPGM(/bin/sh)

TSS PERMIT(DOMCON) IBMFAC(BPX.DAEMON)
                   ACCESS(READ)

TSS ADD(DEPTACID) DSN(DOMCOM.WTO.LOAD)

TSS PERMIT(DOMCON) DSN(DOMCOM.WTO.LOAD)
                   ACCESS(READ)
```

Lotus Notes and Novell Directory Services for z/OS

CA Top Secret can map a user identity from a Lotus Notes or Novell Directory Services for z/OS application to an CA Top Secret ACID. After an application determines a user's CA Top Secret ACID, it can use the ACID to access MVS resources, such as data sets and USS files.

Examples: Mapping Lotus and Novell

This example maps CA Top Secret to a Lotus Notes for z/OS user identity name:

```
TSS ADD(acid) SNAME(lotus user identity name)
```

This example maps CA Top Secret to a Novell Directory Services for z/OS user identity name:

```
TSS ADD(acid) UNAME(Novell Directory Services user identity name)
```

Distributed File Service (DFS)

DFS allows users to access and share files stored on a file server anywhere on the network without knowing the physical location of the file. Files are part of a single namespace. No matter where in the network a user is, the file can be found using the same name. CA Top Secret definitions are required for the DFS and the DFS Client (DFSCM).

Setting up DFS

To set up DFS, enter:

```
TSS CREATE(DFSGRP) TYPE(GROUP)
                   NAME('DFS GROUP')
                   DEPT(dept)

TSS CREATE(DFS) TYPE(USER)
                PASS(PASSWORD,0)
                NAME('DFS region acid')
                DEPT(dept)

TSS ADD(DFSGRP) GID(4)

TSS ADD(DFS) UID(0)
             HOME(/opt/dfslocal/home/dfscntl)
             DFLTGRP(DFSGRP)
             GROUP(DFSGRP)

TSS ADD(STC) PROCN(DFS)
             ACID(DFS)

TSS ADD(STC) PROCN(DFSCM)
             ACID(DFS}
```

Distributed File Server SMB SUPPORT

DFS SMB support provides a server that makes the HFS file available to SMB clients. Server Message Block (SMB) is a protocol for remote file/print access used by Windows and OS/2 clients.

To use SMB support

1. Enter:

   ```
   FAC(USERXX=NAME=DFS)
   FAC(DFS=PGM=DFSCNTL)
   FAC(DFS=NOTSOC,RES,NOIJU,AUTHINIT)
   ```

   A facility for DFS in the TSS control file is defined.

2. Enter:

   ```
   TSS CRE(DFS) TYPE(USER)
                NAME('DFS REGION ACID')
                FAC(BATCH,STC)
                PASS(XXXX,0)
                MASTFAC(DFS)
                NORESCHK
                NODSNCHK

   TSS ADD(DFS) UID(0)
                HOME(/opt/dfslocal/home/dfscntl)
                DFLTGRP(DFSGRP)
                GROUP(DFSGRP)
   ```

   The region ACID for DFS is created and an OMVS segment added.

3. Enter:

   ```
   TSS ADD(STC) PROCN(DFS procname)
                ACID(DFS)
   ```

   The DFS procedure is defined to CA Top Secret.

4. Enter:

   ```
   TSS ADD(STC) PROCN(DFS kern procname)
                ACID(DFS)
   ```

   The DFSKERN procedure is define to CA Top Secret.

SMB ENCRYPTED PASSWORD SUPPORT

To enable the SMB server to use the encrypted password processing with z/OS add the entry DCE.PASSWORD.KEY to the SDT KEYSMSTR record.

This command has the following format:

```
TSS ADD(SDT) KEYSMSTR(DCE.PASSWORD.KEY)
             DCENCRY(kkkkkkkk)
             KEYMASK | KEYNCRY
```

KEYSMSTR

This attribute has only one value, which is DCE.PASSWORD.KEY, entered in uppercase characters.

DCENCRY

Specifies the 16-character hexadecimal encryption key

KEYMASK

Specifies that the DCENCRY key is used to mask the user's DCE password when it is stored in the DCEKEY field of the user's ACID record.

Default: KEYMASK

KEYENCRY

Specifies that the DCENCRY key is used to encrypt the user's DCE password when it is stored in the user's ACID record

Note: Only the MSCA can specify the KEYSMSTR keyword.

Examples: SMB passwords

This example defines the string c1c2c3c4c5c6c7cc8 as the encryption key value for SMB password support:

```
TSS ADD(SDT) KEYSMSTR(DCE.PASSWORD.KEY)
             DCENCRY(C1C2C3C4C5C6C7C8)
```

This example deletes the KEYSMSTR record for the SDT:

```
TSS DEL(SDT) KEYSMSTR(DCE.PASSWORD.KEY)
```

This example replaces the encryption key and use the DES encryption to mask:

```
TSS REP(SDT) KEYSMSTR(DCE.PASSWORD.KEY)
             DCENCRY(0123456701234567)
```

This example lists the current encryption value:

```
TSS LIST(SDT) KEYSMSTR(DCE.PASSWORD.KEY)
```

NFS (Network File System)

z/OS NFS enables remote access to z/OS data sets and USS HFS files and directories. NFS provides the ability to protect file systems on MVS through four protection schemes. This setting is defined within the NFS 'Site Attributes' attribute 'Security'.

Possible settings include:

NONE

Do restrict access. No MVS user ID required.

EXPORTS

Restrict access by client IP address. No SAF check.

SAF

Use SAF to control access to data sets. SAF check executed.

SAFEXP

Use SAF and EXPORTS to control access. SAF check (most secure).

Both SAF and SAFEXP require the user to use the 'mvslogin' process to validate access through a SAF call. CA recommends a minimum of security (SAF). Users who attempt to access HFS data must have a valid OMVS segment assigned to their MVS ACID. Access to HFS files is done by validating the client's UID and group against the file UNIX permission bits. Under normal circumstances access to MVS data sets requires both the z/OS NFS server and client user to pass a security check for the resource. The exception to this is when 'DataCaching' is enabled. DataCaching causes data to be stored on the z/OS NFS client system.

The first user attempting to access an MVS data set must pass a SAF security check. This SAF call is issued by the z/OS NFS Server. Once passed, the data set is stored in the z/OS NFS Client server. Subsequent requests allow all users access to the cached data without further restrictions. Data caching by default is enabled. CA recommends 'DataCaching' be disabled. With DataCaching(N) no client data caching takes place, therefore each user must pass the z/OS NFS Security server check prior to being granted access to data. z/OS NFS Server 'Site Attribute' 'checklist' lists the files and or directories for which SAF security is bypassed even when SAF or SAFEXP is specified. For this reason proper care must be taken to secure this data set. The checklist data set is defined by the CHKLIST DD in the MVSNFS procedure.

CA Top Secret Support for z/OS NFS

To define a facility enter:

```
FAC(USERxx=NAME=NFS)
```

To create ACIDs for started task, create an ACID for each of the four procedures used by NFS: MVSNFS, MVSNFSC, MVSSTATD, and MVSLOCKD.

These ACIDs require an OMVS segment having UID(0). The NFS Server and Client ACIDs require access to data sets to which the remote user accesses. This can be accomplished by explicitly permitting the desired data sets or by adding the NODSNCHK bypass attribute.

1. Enter the commands:

```
TSS CREATE(MVSNFS) NAME('NFS SERVER')
                   DEPT(dept)
                   TYPE(USER)
                   PASS(password,0)
                   FAC(STC)

TSS ADD(MVSNFS) UID(0)
                GROUP(OMVSGRP)
                DFLTGRP(OMVSGRP)
TSS ADD(MVSNFS) NODSNCHK  **or permit all required data sets**
TSS ADD(MVSNFS) MASTFAC(NFS)
TSS ADD(MVSNFS) SOURCE(INTRDR)
```

   The z/OS NFS Server ACID is created

2. Enter the commands:

```
TSS CREATE(MVSNFSC) NAME('NFS CLIENT')
                    DEPT(dept)
                    TYPE(USER)
                    PASS(password,0)
                    FAC(STC)

TSS ADD(MVSNFSC) UID(0)
                 GROUP(OMVSGRP)
                 DFLTGRP(OMVSGRP)

TSS ADD(MVSNFSC) NODSNCHK  **or permit all required data sets**

TSS ADD(MVSNFSC) MASTFAC(NFS)

TSS ADD(MVSNFSC) SOURCE(INTRDR)
```

   The z/OS NFS Client ACID is created.

3. Enter the commands:

```
TSS CREATE(MVSNLM) NAME('NFS NLM')
                   DEPT(dept)
                   TYPE(USER)
                   PASS(password,0) FAC(STC)

TSS ADD(MVSNLM) NODSNCHK  **or permit all required data sets**

TSS ADD(MVSNLM) UID(0)
                GROUP(OMVSGRP)
                DFLTGRP(OMVSGRP)

TSS PER(MVSNLM) DSN( per all required data sets )

TSS ADD(MVSNLM) SOURCE(INTRDR)
```

   The z/OS NFS NLM (Network Lock Manager) ACID is created.

4. Enter the commands:

```
TSS CREATE(MVSNSM) NAME('NFS NSM')
                   DEPT(dept)
                   TYPE(USER)
                   PASS(password,0)
                   FAC(STC)

TSS ADD(MVSNSM) NODSNCHK  **or permit all required data sets**

TSS ADD(MVSNSM) UID(0) GROUP(OMVSGRP) DFLTGRP(OMVSGRP)

TSS PER(MVSNSM) DSN( permit all required data sets )

TSS ADD(MVSNSM) SOURCE(INTRDR)
```

   The OS/390 NFS NSM (Network Status Monitor) ACID is created.

5. Enter the commands:

```
TSS ADD(STC) PROCN(MVSNFS)
             ACID(MVSNFS)

TSS ADD(STC) PROCN(MVSLOCKD)
             ACID(MVSNLM)

TSS ADD(STC) PROCN(MVSSTATD)
             ACID(MVSNSM)

TSS ADD(STC) PROCN(MVSNFSC)
             ACID(MVSNFSC)
```

   The procedures are added to the STC ACID.

Workload Management (WLM)

The WLM ISPF application is protected by a SAF call. Access to the WLM ISPF application is controlled through the definition of a facility class in CA Top Secret. READ or UPDATE access to the entire WLM service definition is the only option available through the CA Top Secret facility access authorization. READ access allows users access to all functions except installing and activating a service definition or policy. When using WLM:

- Specify an access of NONE for the facility resource
- Limit the number of users authorized to read and update the WLM application to those who maintain the WLM policy and performance personnel
- Review the requirement for operations to have access to install a service definition versus activating an existing policy

To authorize the facility for WLM, enter:

```
TSS ADD(deptacid) IBMFAC(MVSADMIN)—Skip this command if already owned

TSS PER(acid) IBMFAC(MVSADMIN.WLM.POLICY) ACC(READ)
```

or

```
TSS PER(acid) IBMFAC(MVSADMIN.WLM.POLICY) ACC(UPDATE)
```

Chapter 5: Digital Certificates

This section contains the following topics:

Digital Certificate Support

A digital certificate is associated to a CA Top Secret user through the user's ACID record, the predefined ACID CERTAUTH, or the predefined ACID CERTSITE.

When using certificates:

- More than one certificate can be added to a user's ACID record.
- Each certificate is unique to a particular user, a certificate cannot be added to more than one ACID unless the certificate is attached to a key ring.
- To define many digital certificates, you may need to define a VSAM file to hold the records. For information, see the *Installation Guide*.

Key Rings

A key ring is a collection of digital certificates associated with an individual user. Once a user has had their identity verified to a system by a certificate that is unique to the user, the user can access additional resources through the certificates on their key ring. Key rings provide an installation-wide method to share digital certificates across multiple servers. A user can have more than one key ring.

Certificate Associations

CA Top Secret lets you check if a certificate has already been added to the CA Top Secret Security File and what ACID it is associated with. Once a certificate has been added to CA Top Secret it can be exported to a new data set.

Certificate Administration Commands

Use the following TSS commands to administer certificates:

ADD and REPLACE

Adds or replaces a certificate label or the START, FOR, UNTIL, TRUST, HITRUST, and NOTRUST parameters.

CHKCERT

Determines if a certificate has been added to the CA Top Secret security file and with what ACID it is associated.

EXPORT

Exports a certificate from CA Top Secret to a new data set.

GENCERT

Generates a certificate from CA Top Secret and adds it to a user.

GENREQ

Generates a PKCS#10 base 64-encoded digital certificate request and writes it to a data set.

REMOVE

Removes the certificate from the user.

For information, see the *Command Functions Guide*.

Key Size

Use the KEYSIZE keyword to specify the size of the private encryption key in decimal bits. The larger the keysize, the more CPU time it takes to generate the certificate. The following table details the requirements for private keys and their size:

Private Key Type	Maximum Key Size
ICSF RSA Key	1024
Non-ICSF DSA Key	2048
Non-ICSF RSA Key	4096
PCICC	4096

Administrator Rules

CA Top Secret administrators:

- Must have ACID(MAINTAIN) for users within their scope, plus MISC4(authority levels)
- Must be defined with an OMVS segment UID, Group, and Default Group to perform any digital certificate commands and USS must be active

CPF Limitations

Any command containing a DIGICERT, KEYRING, SERIALNM, ISSUERDN, LABLCERT, or LABLRING keyword is not propagated.

The CHKCERT, EXPORT, GENCERT, GENREQ, REKEY, RENEW, and ROLLOVER functions cannot be propagated because the certificate generated on the CPF connected system would not be the same as the one on originating system. For the certificates to be the same, export the DSN and then do an ADD of that DIGICERT.

CPF Limitations

Some digital certificate commands are available for use with the Command Propagation Facility (CPF). However, the following CPF limitations exist for digital certificate commands.

Note: For complete information about CPF, see the *CA Top Secret User Guide*.

- Only the following certificate-related commands are sent through CPF:

  ADD

  Adds or replaces a certificate LABEL or the START, FOR, UNTIL, TRUST, HITRUST, and NOTRUST parameters.

  REMOVE

  Removes the certificate from the user.

  REPLACE

  Replaces a certificate LABEL or the START, FOR, UNTIL, TRUST, HITRUST, and NOTRUST parameters.

- The following keywords, issued by ADD, REMOVE, and REPLACE, are sent through CPF:
  - DIGICERT
  - ISSUERDN
  - KEYRING
  - LABLCERT
  - LABLRING
  - SERIALNM
- The following certificate-related commands are *not* sent through CPF:
  - CHKCERT
  - EXPORT
  - GENCERT
  - GENREQ
  - REKEY
  - ROLLOVER

PKCS #11 Tokens

PKCS #11 is the cryptographic token interface standard. It specifies an application programming interface (API) to tokens, that hold cryptographic information and perform cryptographic functions.

z/OS PKCS #11 tokens are created using system software such as CA Top Secret, ACF2, RACF, the gskkyman utility, or by applications using the C API.

Each token has a unique token name, or label, specified by the end user or application when the token is created.

A token name must follow these rules:

- Up to 32 characters in length
- Permitted characters are:
  - Alphanumeric
  - National: @ (X'5B'), # (X'7B'), or $ (X'7C')
  - Period
- The first character must be alphabetic or national
- Lowercase letters can be used but are folded to uppercase

To use PKCS #11 tokens with CA Top Secret, use the TSS P11TOKEN command function with the BIND, IMPORT, TOKENADD, TOKENDEL, TOKENLST and UNBIND keywords.

For information, see the *Command Function Guide*.

Token Access Control

The PKCS #11 standard is for systems that grant access to token information based on a personal identification number (PIN).

The standard defines two types of users, each has its own PIN:

Standard User (User)

The User has access to the private objects on a token and has the power to change their PIN. The User cannot reinitialize a token. The PIN the user enters determines which role that user takes. z/OS does not use PINs, profiles in the SAF CRYPTOZ class control access to tokens.

Security Officer (SO)

The SO can initialize a token (zero the contents) and set the User's PIN. The SO can access the public objects on the token but not the private ones.

A user can fill both roles by knowing both PINs.

Resources

For each token, there are two resources in the CRYPTOZ class:

USER.token-name

Controls the access of the User role to the token.

SO.token-name

Controls the access of the SO role to the token.

A user's access level to each of these resources (read, update, or control) determines the user's access level to the token.

For information on the CRYPTOZ class, see the IBM guide C*ryptographic Services Integrated Cryptographic Services Facility Writing PKCS #11 Applications SA23-2231*.

Example: Controlling access to z/OS PKCS#11 tokens

In this example, a company uses z/OS PKCS #11 tokens as the key stores for their FTP and Web servers. The company naming convention is that all tokens have the owning ACID as the high-level qualifier. The owning ACID the FTP is the daemon FTPSERV and Web server is the daemon WEBSERV. User01 is the administrator for the servers.

The security administrator (SECADMIN) creates the protection profiles for the tokens. His goal is to give user01 the Security Officer role for these profiles, and to give the daemon ACIDs the User role.

The security administrator:

- Creates profiles for the security officer's access to the FTP and Web Server tokens:

  ```
  TSS ADD(SECDEPT) CRYPTOZ(SO.FTPSRV.*)
  TSS ADD(SECDEPT) CRYPTOZ(SO.WEBSRV.*)
  ```

- Creates profiles for the standard user's access to the FTP and Web Server tokens:

  ```
  TSS ADD(SECDEPT) CRYPTOZ(USER.FTPSRV.*)
  TSS ADD(SECDEPT) CRYPTOZ(USER.WEBSRV.*)
  ```

- Gives user01 Strong SO power for the tokens by giving her CONTROL access to the profiles that protect the tokens:

  ```
  TSS PERMIT(user01) CRYPTOZ(SO.FTPSRV.*) ACCESS(CONTROL)
  TSS PERMIT(user01) CRYPTOZ(SO.WEBSRV.*) ACCESS(CONTROL)
  ```

- Gives the users FTPSRV and WEBSRV Weak User power for their respective tokens. This allows them to use the private objects within the tokens, but not change the set of trusted CA certificates.

  ```
  TSS PERMIT(FTPSRV) CRYPTOZ(USER.FTPSRV.*) ACCESS(UPDATE)
  TSS PERMIT(WEBSRV) CRYPTOZ(USER.WEBSRV.*) ACCESS(UPDATE)
  ```

When complete:

- User01 has Strong SO power over the tokens for the FTP server and the Web server and can create the required tokens
- FTPSERV and WEBSRV have User power over their respective tokens and can use them as key stores

PKCS #11 Functions Audit

PKCS #11 functions are audited with:

SMF type 82 subtype 1

Written during ICSF initialization. Contains the data set name of the token data set (TKDS).

SMF type 82 subtype 21

Written when a member joins or leaves a sysplex group contains the cryptographic keys data set (CKDS) data set name if the member joined or left the ICSF CKDS sysplex group, or the TKDS data set name if the member joined or left the ICSF TKDS sysplex group.

ICSF writes SMF type 82 subtype 23

Written whenever a TKDS record for a token or token object is created, modified, or deleted. ICSF does not write SMF records for changes to session objects.

PKCS 7 and PKCS 12 Certificate Processing

If a PKCS 7 certificate package contains more than one certificate, the product considers all certificates except the first certificate to be certificate authority (CA) certificates. For a PKCS 12 certificate package, any certificate that lacks a “local key ID” is considered a CA certificate.

The product sorts the CA certificates (to determine the hierarchy) and then inserts the certificates in hierarchical order under the CERTAUTH ID so that each certificate in the package can be verified using its previously inserted signing certificate. The inserted CA certificates have a record ID in AUTO*nnnn* format, where *nnnn* is a number between 0001 and 9999. The product also inserts the end-entity certificate.

Note: The highest-level CA certificate will not necessarily have an AUTO*nnnn* number less than the other CA certificates being inserted.

A CA certificate that is already known to CA Top Secret retains trust status. If an error occurs during the addition of certificates from a certificate package, the product does not perform CERTAUTH assignments for certificates that were already added.

General Rules

A certificate generated by the GENCERT command used for SSL server authentication must be exported to the client's repository. The public key must be available to decrypt the server's certificate during the SSL authentication handshake.

Client software might be PC workstation, Internet browser, AS400, Windows NT, MQSeries, FTPSSL, or QWSSSL. They also need authority to the IBMFAC.

To establish IBMFAC authority

1. Enter the command:

   ```
   TSS ADD(tssdept) IBMFAC(IRR)
   ```

   The IBMFAC is owned.

2. Enter the command:

   ```
   TSS PERMIT(tssadmin1) IBMFAC(IRR.DIGTCERT.LISTRING)
                         ACCESS(UPDATE)
   ```

   Permission is applied to the administrator.

If the administrator submits batch scripts for certificates, they must include REGION=0M in their job statement within the JCL.

Third Party Vendor Certificate Registration

Any certificate obtained from third party vendors, such as Verisign, can be registered to CA Top Secret via the TSS ADD command. Once the certificate is received from the vendor it must be placed into a cataloged MVS data set so that it can be accessed by CA Top Secret. This data set would thereby represent the value specified in the DCDSN keyword of the TSS ADD command.

To register a third part certificate to CA Top Secret, enter the command:

```
TSS ADD(name) DIGICERT(namecert)
              DCDSN(name.certificate.data)
              TRUST
```

Add a Certificate to an ACID

When you add a digital certificate, the DIGICERT and DCDSN keywords are required. All other keywords are optional.

To add a certificate to an ACID, enter the following command:

```
TSS ADDTO(acid|CERTAUTH|CERTSITE) DIGICERT(name)
                                  DCDSN(dataset_name)
                                  [CERTNSER(nnnnnnnnnnnnnnnn)]
                                  [START(sdate)]
                                  [FOR(ddd)|UNTIL(date)]
                                  [LABLCERT(label name)]
                                  [LABLPKDS(PKDS-label|*]
                                  [TRUST|NOTRUST|HITRUST]
                                  [ICSF|PCICC|DSA]
                                  [PKCSPASS('PKCSPASS PASSWORD')]
```

acid

Specifies a user ACID.

CERTAUTH

Specifies an ACID in which your installation can maintain certificates that were generated by a third party certificate authority (CA). This ACID is predefined in CA Top Secret. You *cannot* add a keyring to this ACID.

CERTSITE

Specifies an ACID in which your installation can maintain site-generated certificates. This ACID is pre-defined in CA Top Secret. You *cannot* add a keyring to this ACID.

DIGICERT

Specifies a case-sensitive character ID that identifies the certificate with the user ACID.

Range: 1 to 8

DCDSN

Specifies the MVS data set containing the digital certificate. The data set must be defined as physical sequential (DSORG=PS) and variable blocked data set (RECFM=VB). The data set name is entered as a fully qualified name without enclosed quotes (LREC=84).

The certificate contained in the data set must be BER-encoded, PKCS-7 BER-encoded, or Privacy Enhanced Mail (PEM)-encoded. PEM certificates must be transported to MVS as TEXT; the other formats must be transported as BINARY.

In addition to the end-entity certificate specified on the ADD command, the product adds all certificate authority (CA) certificates contained in a PKCS 7 or PKCS 12 certificate package. These certificate packages generally contain an end-entity user certificate and a chain of CA certificates. The trust status of the first added CA certificate takes the value specified on the Insert command. The other added CA certificates take the trust value of the signing certificate.

For PKCS 7 and PKCS 12 certificate package, a certificate that meets any of the following criteria is added with a trust status of NOTRUST:

- The certificate is expired or has a validity period that is not completely within the validity period of its signing certificate.
- The signing certificate of the certificate is not in a PKCS 7 or PKCS 12 package or not in CA Top Secret.

If a CA certificate is already known to CA Top Secret, the certificate retains its trust status. Each added CA certificate receives a label of AUTO*xxxx*, where *xxxx* is an available number between 1 and 9999.

Note: If an error occurs during the addition of certificates from a PKCS 7 or PKCS 12 certificate package, the product does *not* remove any CERTAUTH certificates that were already added.

Certificates containing unsupported critical extensions *cannot* be inserted into CA Top Secret. Noncritical extensions are ignored. Supported critical extensions are as follows:

- keyUsage - 2.5.29.15
- basicConstraints - 2.5.29.19
- subjectAltname - 2.5.29.17
- issuerAltname - 2.5.29.18
- certificatePolicies - 2.5.29.32
- policyMappings -2.5.29.33
- policyConstraints - 2.5.29.36
- nameConstraints - 2.5.29.30
- extKeyUsage - 2.5.29.37
- subjectKeyIdentifier - 2.5.29.14

- authorityKeyIdentifier - 2.5.29.35
- hostIdMapping - 1.3.18.0.2.18.1

Ranges: The data set must be cataloged and can be up to 44 characters long. The length of the serial number and certificate authority distinguished name must be less than 246.

CERTNSER

Specifies the hex value of the next serial number used by this certificate to sign another certificate. Every byte must be specified, including leading zeros.

Size: 16 bytes

START

Specifies an optional activation date. This date is not the same as the activation date defined in the certificate itself. The web server validates that date. This date gives the security administrator the ability to specify when the certificate will become active on MVS.

FOR|UNTIL

Specifies an optional expiration date. This date is not the same as the expiration date defined in the certificate. The web server validates that date. This date gives the security administrator the ability to specify when the certificate will expire on MVS.

LABLCERT

Specifies the label to be associated with the certificate being added to the user. Spaces are allowed if you use single quotation marks. This label is used as an identifier (instead of the serial number and issuer's distinguished name) and must be unique for the individual user. If you do not specify a label, the label field defaults to the value specified within the DIGICERT keyword.
Range: Up to 32 characters

HITRUST|TRUST|NOTRUST

Specifies a trust status for the certificate:

HITRUST

Specifies that the certificate is highly trusted and trusted. Any certificate usage applying to trusted certificates applies to highly trusted certificates. However, only certificate authority certificates (CERTAUTH) can be highly trusted.

TRUST

Specifies that the certificate is trusted, which means the certificate is valid for the user, site, or certificate authority, and the private key has not been compromised. Trusted user certificates can be used to authenticate a user ID. Trusted CERTSITE certificates can be used without authentication. Trusted CERTAUTH certificates can be used to authenticate other certificates

NOTRUST

Specifies that the certificate is not trusted.

If a trust status is not specified, the product determines status as follows:

- If the certificate's signature can be verified, the certificate has not expired, and the certificate's validity dates fall within the range of the signing certificate, the trust status is set to the trust status of the certificate authority.
- The default trust status for self-signed certificates is TRUST.
- If the certificate has expired, has an invalid validity date range, or cannot be verified, the trust status is set to NOTRUST.
- If the trust status is coming from the signing certificate and the signing certificate has HITRUST, the status of the new certificate becomes TRUST.
- If the certificate was signed by another certificate, the status is set to TRUST if the following conditions are met:
  - The signing certificate can be located in the database.
  - The signing certificate is a certificate authority certificate (CERTAUTH).
  - The signing certificate is not expired.
  - The signing certificate's signature is valid.
  - The validity dates of the certificate being added fall within the range of the signing certificate's validity dates.

If the product cannot determine status through other methods, the certificate is inserted as not trusted (NOTRUST) with a message that explains the reasons.

Note: If the signing certificate's signature is invalid, the certificate is not inserted.

LABLPKDS(*PKDS_label*|*)

(Optional) Specifies the PKDS label of the record created in the ICSF Public Key Data Set (PKDS). The field can be used with the ICSF, PCICC, NISTECC, or BPECC, but many of these keywords cannot be used together (see individual keyword descriptions for details). If neither ICSF or PCICC is specified, a PCICC key is generated by the hardware and saved in CRT format in the ICSF PKDS. If NISTECC or BPECC is specified, an ECC key is generated; otherwise, an RSA key is generated.

Specify (*) to take the value from the LABLCERT keyword. In that case, LABLCERT is specified alongside LABLPKDS(*).

The PKDS label must conform to ICSF label syntax rules. The first character must be alphabetic or national. The field is folded to uppercase.

A key pair is not generated because the key is taken from the certificate in the data set. If the data set contains a PKCS 12 package, the private key is placed in the ICSF PKDS (with the format being determined by the ICSF or PCICC keywords). If ICSF is also specified, the private key is stored in the ICSF PKDS as an ICSF RSA Modulus-Exponent (ME) key token. If PCICC is specified, the private key is stored as an ICSF RSA Chinese Remainder Theorem (CRT) key token. If the private key has a bit size greater than 1024, PCICC must be specified.

If the data set contains a single certificate or a PKCS 7 chain, the public key is placed in the ICSF PKDS in RSA public key format. If this insertion results in an error (from attempting to insert a record with the same PKDS label as a record that already exists in the PKDS), the product reads the existing record. If that record contains a private key, and the public key being inserted corresponds to the private key, the CA Top Secret record is updated to indicate that a private key exists for the record.

Valid characters: Alphanumeric characters, national (@,#,$) characters, or a period(.).

Limits: Up to 64 characters

ICSF

Specifies to store the private key in the ICSF data facility. The IBM ICSF feature provides an interface to the cryptographic hardware on z/OS. You must have cryptographic hardware installed and enabled on your system. If ICSF, PCICC, or LABLPKDS is not specified with ADD, the key is stored in the security file as a non-ICSF key.

Note: ICSF is valid to create only RSA keys with a length up to 1024 bytes.

PCICC

(Optional) Specifies that the key pair is generated using the PCI Cryptographic Coprocessor and that the private key is stored in ICSF. When PCICC is not specified, the key pair is generated using software. PCICC cannot be used with the DSA, DSN, or ICSF parameters.

If a PCI cryptographic coprocessor is not present or operational or if ICSF is not active or configured for PKA operations, an error message is displayed and processing terminates. If ICSF, PCICC, or LABLPKDS is not specified, the key pair is generated using software and stored in the security file as a non-ICSF key.

PKCSPASS

Specifies the password used to decrypt the PKCS #12 certification package. This password must conform to PKCS 12 standards and must be the same as the password that was specified when the certificate was exported. The password may be mixed case and up to 255 bytes.

Note: A password can be specified only with a PKCS #12 certificate. The product supports only PKCS #12 certificates that adhere to the PKCS #12 v1.0 standard published by RSA. These certificates are defined with a 3 in the version number of the PKCS #12 certificate package.

Example: Use DCDSN to Add a Certificate

This example uses DCDSN:

```
TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER01.CERTIF.001)
```

Example: Use START to Add a Certificate

This example uses START:

```
TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER01.CERTIF.001)
                START(10/01/03)
```

Example: Use FOR|UNTIL to Add a Certificate

These examples use FOR and UNTIL:

```
TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER01.CERTIF.001)
                FOR(30)

TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER01.CERTIF.001)
                UNTIL(10/01/03)
```

Example: Use LABLCERT to Add a Certificate

This example uses LABLCERT:

```
TSS ADD(USER01) DIGICERT(DIGI0001)
                LABLCERT('label for digicert 001')
                TRUST|NOTRUST | HITRUST
```

Example: Use TRUST|NOTRUST|HITRUST to Add a Certificate

These examples use TRUST, NOTRUST, and HITRUST:

```
TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER01.CERTIF.001)
                TRUST

TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER01.CERTIF.001)
                NOTRUST

TSS ADD(CERTAUTH) DIGICERT(DIGI0001)
                   DCDSN(USER01.CERTIF.001)
                  *HITRUST
```

Example: Use ICSF to Add a Certificate

This example uses ICSF:

```
TSS ADD(USER01) DIGICERT(DIGI0001)
                DCDSN(USER04.CERTIF.001)
                ICSF
```

Certificate Generation

Use the GENCERT command to create a digital certificate and potentially a public/private key pair.

The DIGICERT keyword is required. If both DCDSN and SUBJECTN are specified, the SUBJECTN information overrides the request data set name. If SUBJECTN is specified, only one of the SUBJECTN sub fields is required.

To generate a certificate, enter the command:

```
TSS GENCERT (CERTAUTH|CERTSITE|acid]
            DIGICERT(8-byte-name)
            [DCDSN(request-data-set-name)\]
            [SUBJECTN ('CN="common-name"
                       T="title"
                       OU="organizational-unit-name1,
                          organizational-unit-name2"
                       O="organizational-name"
                       L="locality"
                       ST="state-or-province"
                       C="2-digit-only country code"')]
            [ALTNAME('IP=numeric-IP-address DOMAIN=internet-domain-name
                     EMAIL=email-address
                     URI=universal-resource-identifier')]
            [ICSF|PCICC|DSA|NISTECC|BPECC]
             [FROMICSF(label-name)]
            [KEYSIZE(key-size)]
            [KEYUSAGE('HANDSHAKE DATAENCRYPT DOCSIGN CERTSIGN KEYAGREE')]
            [LABLCERT(label-name)]
            [LABLPKDS(PKDS-label|*]
            [NBDATE(mm/dd/yy) NBTIME(hh:mm:ss)]
            [NADATE(mm/dd/yy) NATIME(hh:mm:ss)]
            [SIGNALG{SHA1|SHA256)]
            [SIGNWITH(acid,digicert)]
```

ACID

A user ACID.

CERTAUTH

Is an ACID in which your installation can maintain certificates that were generated by a third party certificate authority (CA). This ACID is pre-defined in CA Top Secret. You cannot add a KEYRING to this ACID.

CERTSITE

An ACID in which your installation can maintain site-generated certificates. This ACID is pre-defined in CA Top Secret. You cannot add a KEYRING to this ACID.

DIGICERT

Specifies a case sensitive character ID that identifies the certificate with the user ACID. The DIGICERT *must* be entered as part of all GENCERT commands since this keyword indicates the name to be used in the digital certificate.
Range: 1 to 8

DCDSN(request-data-set-name)

Specifies the name of an optional data set that contains the PKCS#10 certificate request data. The request data set name can be the output from a TSS GENREQ command. The request data contains the user's generated public key and X.509 distinguished name. The request data must be signed, DER-encoded, and then Base64 encoded according to PKCS#10 standard. The data set must be cataloged.

If DCDSN is specified, CA Top Secret does not generate a key pair (meaning private and public key) because this data set contains the user's public key. SIGNWITH *must also be specified* because the request-data-set-name (in DCDSN) does not contain a private key.
Range: Up to 44 characters

SUBJECTN

You can use A-Z and 0-9. The only exception is C=COUNTRY. This is a 2-digit value field. If DCDSN or SUBJECTN is not specified, the SUBJECTN will default to the ACID name field.

```
[SUBJECTN ('CN="common-name"
            T="title"
            OU="organizational-unit-name1,organizational-unit-name2"
            O="organizational-name"
            L="locality"
            ST="state-or-province"
            C="2-digit-only-country code"')]
```

Notes:

- If any of the values contain blanks, they must be enclosed in double quotes.
- The complete SUBJECTN phase must be enclosed in parenthesis and single quotes.
- The attributes are separated by spaces. No matter how many spaces there are between attributes, they count as one space.
- Each attributes has a limit of 64 characters.
- Multiple values can be specified for the Organizational Unit (OU=).

Range:

- Up to 229 characters for a self-signed certificate if SDNSIZE(255) control option is specified, 1007 characters if SDNSIZE(1024) control option is specified
- Up to 255 characters for a non self-signed certificate if SDNSIZE(255) control option is specified, 1024 characters if SDNSIZE(1024) control option is specified

- Up to 229 characters for a certificate added to CERTAUTH if SDNSIZE(255) control option is specified, 1007 characters if SDNSIZE(1024) control option is specified

ALTNAME

Specifies the appropriate values for the SubjectAltname extension, of which one or more values might be coded. *There is no default.* The following are possible values that can be used:

IP

Specifies a string containing a fully qualified numeric IP address in:

- IPv4 dotted decimal format—four decimal numbers between 0 and 255 separated by periods. For example:

  `141.202.1.255`

- IPv6 format—eight parts divided by colons with each part a hexadecimal number between 0 and FFFF. For example:

  `1080:23B4:324:4:3BCD:26:39F4:332`

- IPv4 compatible IPv6 address—a combination of the two, six parts of the IPv6 followed by the IPv4 address. For example:

  `0:0:0:0:0:FFFF:141.202.1.255`

The maximum field size is 45 bytes

DOMAIN

Specifies a string containing a fully qualified internet domain name.

For example: ALTNAME(DOMAIN=CA.COM)

EMAIL

Specifies a string containing a fully qualified email address.

For example: ALTNAME(EMAIL=JAMES@Kingdom.net)

URI

Specifies the universal resource identifier.

For example: ALTNAME(URI=WWW.CA.COM)

Notes:

When you specify multiple parameters to ALTNAME, you must include one single quote at the beginning and end of parameter list.

For example: ALTNAME('IP=201.100.10.9 EMAIL=my.email@test.net')

Multiple parameters are separated with a space (see example above).

ICS

If ICSF is specified and the IBM ICSF feature is enabled, the private key is stored in the ICSF data facility.

PCICC

(Optional) Specifies that the key pair is generated using the PCI Cryptographic Coprocessor and that the private key is stored in ICSF. When PCICC is not specified, the key pair is generated using software. PCICC cannot be used with the DSA, DSN, or ICSF parameters.

If a PCI cryptographic coprocessor is not present or operational or if ICSF is not active or configured for PKA operations, an error message is displayed and processing terminates. If ICSF, PCICC, or LABLPKDS is not specified, the key pair is generated using software and stored in the security file as a non-ICSF key.

DSA

(Optional) Specifies that the key pair is generated using the Digital Signature Algorithm instead of the RSA algorithm. The DSA algorithm creates key pairs that can be only used for signing data. The RSA algorithm creates key pairs that can be used to sign data and to encrypt data. This parameter cannot be used in conjunction with the ICSF, PCICC, NISTECC, or BPECC parameters. When specifying the DSA parameter, the KEYSIZE parameter can be as high as 2048.

NISTECC

(Optional) Specifies the key pair should be generated using National Institute of Standards and Technology (NIST) algorithm instead of the RSA algorithm. This parameter cannot be used with the ICSF, DSA, or BPECC parameters.

BPECC

(Optional) Specifies to generate the key pair using the brainpool ECC algorithm instead of the RSA algorithm. This parameter cannot be used with the ICSF, DSA, or NISTECC parameters.

FROMICSF

(Optional) Specifies that the public key for this certificate will be obtained from ICSF using the specified PKDS label. The private key of the source certificate, if one exists, will not be associated with the new certificate. FROMICSF cannot be specified with the DCDSN or LABLPKDS parameters. SIGNWITH must be specified when FROMICSF is specified.

KEYSIZE

- Specifies the size of the private encryption key in decimal bits.
- **Limits:**
  - Non-NISTECC and non-BPECC: any numeric value from 512 to 4096.
  - For NISTECC, one of the following: 192, 224, 256, 384, 521.
  - For BPECC, one of the following: 160, 192, 224, 256, 320, 384, 512.
- **Default:**
  - Non-NISTECC and non-BPECC:1024.
  - NISTECC and BPECC: 192.

The maximum key size is dependent on the private key type.

Private key type maximum key sizes are:

- BPECC key—512 bits
- NISTECC key—521 bits
- ICSF RSA key—1024 bits
- DSA key —2048 bits
- Non-ICSF RSA key—4096 bits
- PCI-class cryptographic coprocessor RSA key—4096 bits

Shorter ECC keys have key strengths comparable to longer RSA keys. The following table displays the comparable strength of each key type:

RSA Key Size (in bits)	NISTECC Key Size (in bits)	BPECC Key Size (in bits)
1024	192	160 or 192
2048	224	224
3072	256	256 or 320
7680	384	384
15360	521	512

Currently, the standard key sizes for RSA keys are as follows:

- 512—Specifies a low-strength key
- 1024—Specifies a medium-strength key
- 2048—Specifies a high-strength key
- 4096—Specifies a very high-strength key

KEYUSAGE

Specifies key attribute information, including the appropriate values for the KeyUsage certificate extension, of which one or more of the values might be coded. For certificate authority certificates (CERTAUTH) the default is CERTSIGN and is always set. There is no default for certificates that are not certificate-authority certificates. Valid values for KEYUSAGE include:

HANDSHAKE

Facilitates identification and key exchange during security handshakes, such as SSL, which set the digital signature and key encipherment indicators. When the key pair is generated with the DSA algorithm, only the digitalSignature bit is set because the keys cannot be used for encryption.

DATAENCRYPT

Encrypts data, which sets the data encipherment indicator. When the key pair is generated using the DSA algorithm, you cannot use the DATAENCRYPT keyword in the Keyusage parameter.

DOCSIGN

Specifies a legally-binding signature, which set the non-repudiation indicator.

CERTSIGN

Specifies a signature for the other digital certificates and CRLs, which sets the keyCertSign an cRLSign indicators.

Note: Include single quotes if specifying more than one value with KEYUSAGE. For example:

```
KEYUSAGE('HANDSHAKE DATAENCRYPT')
```

KEYAGREE

Facilitates key exchange, which sets the keyAgreement indicator. This usage is valid only for NISTECC and BPECC keys. A certificate with no keyUsage value other than keyAgreement cannot be used for signing.

LABLCERT

Specifies an optional and case-sensitive label to be associated with the certificate being added to the user. Spaces are allowed if you use single quotes. This label is used as a handle instead of the serial number and issuer's distinguished name, and must be unique for the individual user. If a label is not specified, the label field will default to the value specified within the DIGICERT keyword.
Range: Up to 32 characters

LABLPKDS(*PKDS—label—name/)**

(Optional) Specifies the PKDS label of the record created in the ICSF Public Key Data Set (PKDS). The field can be used with the ICSF, PCICC, NISTECC, or BPECC, but many of these keywords cannot be used together (see individual keyword descriptions for details). If neither ICSF or PCICC is specified, a PCICC key is generated by the hardware and saved in CRT format in the ICSF PKDS. If NISTECC or BPECC is specified, an ECC key is generated, otherwise an RSA key is generated.

Specify (*) to take the value from the LABLCERT keyword. In that case, LABLCERT is specified along side LABLPKDS(*). If LABLPKDS(*) is specified without the LABLCERT keyword, an error message is displayed.

In either case, the PKDS label must conform to ICSF label syntax rules. The first character must be alphabetic or national. The field is folded to uppercase.

Valid characters: Alphanumeric, national (@,#,$) or period(.).

Limits: Up to 64 characters

NADATE/NATIME

Specifies the effective dates and times to not be used in the digital certificate. The NADATE specifies the “not after” date after which a digital certificate cannot be used. The NATIME specifies the “not after” time after which the certificate cannot be used. The certificate is deactivated after this date and time.

Date and time fields are optional, except if time is specified, date is required. If NADATE is omitted, the default is one year from the date the certificate is generated.

If an expire date is not also specified, the NBDATE year specified must fall within the range 1950-2048, since the NADATE date defaults to the active day and time plus one year.

The certificate DATE FORMAT are not govern by the DATE parm in CA Top Secret. The format will be MM/DD/YY.

Year Range: 1950-2049

NBDATE/NBTIME

Specifies the effective dates and times to be used in the digital certificate. The NBDATE specifies the *not before* date which a digital certificate can be used. The NBTIME specifies the *not before* time which the certificate can be used. The certificate is activated at the specified date and time.

Date and time fields are optional, except if time is specified, date is required.

If an expire date is not also specified, the NBDATE year specified must fall within the range 1950–2048, because the NADATE date defaults to the active day and time plus one year.

The certificate DATE FORMAT is not governed by the DATE parameter in CA Top Secret. The format is MM/DD/YY.

Year Range: 1950–2049

SIGNALG

(Optional) Specifies the digital certificate signing algorithm to be used when generating a new certificate. Possible values are SHA1 and SHA256.

Default: SHA1 or SHA256 for RSA certificates when key size is 2048 or larger.

Note: SHA256 cannot be used when DSA is specified.

SIGNWITH

Specifies the certificate with a private key that is signing the certificate. If not specified, the default is to sign the certificate with a private key of the certificate that is being generated. This creates a self-signed certificate. 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 generated and processing stops. If DCDSN is specified on the GENCERT command, the SIGNWITH keyword is required.

Self-signed certificates are always trusted, while all other certificates are created with the trust status of the certificate specified in the SIGNWITH keyword. If the certificate specified in the SIGNWITH keyword is not trusted, an informational message is issued, but the certificate is still generated.

The length of the SUBJECTN field on the certificate specified for SIGNWITH cannot exceed 229 if SDNSIZE(225) is specified, or 1007 if SDNSIZE(1024) is specified.

Examples: GENCERT Command

This example uses DIGICERT:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
```

This example uses DCDSN:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERTIF.001)
```

This example uses the ICSF attribute:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    ICSF
```

This example uses PCICC:

```
TSS GENCERT(USER01) DIGICERT(DIGI001)
                    PCICC
```

This example uses NISTECC:

```
TSS GENCERT(USER01) DIGICERT(DIGI001)
                    NISTECC
```

This example uses BPECC:

```
TSS GENCERT(USER01) DIGICERT(DIGI001)
                    BPECC
```

These examples uses KEYSIZE:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERTIF.001)
                    KEYSIZE(512)
```

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERTIF.001)
                    KEYSIZE(768)
```

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERTIF.001)
                    KEYSIZE(1024)
```

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERTIF.001)
                    KEYSIZE(2048)
                    PCICC|DSA

TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERTIF.001)
                    KEYSIZE(4096)
```

This example uses NADATE/NATIME:

```
TSS GENCERT(user1) DIGICERT(cert0001)
                   DCDSN(user1.cert.data)
                   NADATE(09/01/03)
                   NATIME(00:00:01)
```

This example uses SIGNWITH:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERT.DATA)
                    SIGNWITH(USER02,CERT002)
```

This example uses SUBJECTN:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    SUBJECTN('CN=“Ted User” ST=NJ')
```

This example uses FROMICSF:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    FROMICSF(USERX.CERT.LABEL)
                    SIGNWITH(USER02,CERT002)
```

This example uses SIGNALG:

```
TSS GENCERT(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER01.CERT.DATA)
                    KEYSIZE(1024)
                    SIGNALG(SHA256)
```

List Digital Certificate Information

To list information about a digital certificate, you can identify the digital certificate through the following information:

- Certificate name or label
- Certificate serial number and issuer's distinguished name
- Certificate segment data

To list the information about a certificate, enter the following command:

```
TSS LIST(acid|CERTAUTH|CERTSITE) [LABLCERT('label_name')]
                [DIGICERT(name)]
                [SERIALNUM(serial_number)]
                [ISSUERDN(issuer_distinguished_name)]
                [SEGMENT(certdata)]
                [SEGMENT(ALL)]
                [KEYRING(name)]
                [LABLRING(name)]
                [CHAIN]
```

For each certificate, the command displays the following information:

- Serial number
- Issuer's distinguished name
- Label
- Status
- Validity dates
- Private key size (If private key is present)
- Private key type (if private key is present) or NONE (if there is no private key)
- PKDS label (if private key is present)
- Keyrings (if private key is present)
- Keyusage
- Altname
- Subject's name as found in the certificate itself (up to 255 bytes if the SDNSIZE(255) control option is specified or up to 1024 bytes if the SDNSIZE(1024) control option is specified)
- Certificate flags

  A "GENREQ" flag specification indicates that the CERTDATA record has been the target of a GENREQ command. We recommend *not* using the REPLACE command to turn on this indicator. REMOVE processing and ROLLOVER (NEWLABLC or NEWDIGIC) processing will not work against certificates that have the GENREQ indicator on. You can bypass the check for GENREQ by specifying the FORCE operand of either command. You can turn off the GENREQ indicator in the following situations:

  - You specified GENREQ when REKEY was intended.
  - You specified REKEY when GENREQ was intended.

  To turn off the GENREQ indicator, specify REPLACE(*acid*) CERTFLAG(NOGENREQ).

  During normal processing, inserting a signed certificate over the GENREQ'd certificate turns off the GENREQ indicator.

Example: List All ACIDs and Associated Digital Certificates

This example lists all ACIDs and the digital certificates associated with them on a system:

```
TSS LIST(ACIDS) DIGICERT(ALL)
```

Example: List All ACIDs and Associated Keyrings

This example lists all ACIDs and the keyrings associated with them on a system:

```
TSS LIST(ACIDS) KEYRING(ALL)
```

Example: List Associated SEGMENT Information for an ACID

This example lists the associated SEGMENT information for a specific ACID:

```
TSS LIST(USER01) SEGMENT(CERTDATA)

TSS LIST(USER01) SEGMENT(RINGDATA)

TSS LIST(USER01) SEGMENT(ALL)
```

Example: List the Associated DIGICERT for an ACID

This example lists the associated DIGICERT for a specific ACID. The command must contain the name of the DIGICERT or KEYRING already associated with the ACID:

```
TSS LIST(USER01) DIGICERT(CERT001)
```

or

```
TSS LIST(USER01) KEYRING(ACCTRING)
```

Example: List the Digital Certificates Associated with an ACID

This example lists the digital certificates associated with an ACID. The command must contain the name of the ACID:

```
TSS LIST(USER01) DIGICERT(ALL)
```

Example: List All Keyrings Associated with an ACID

This example lists all keyrings associated with an ACID. The command must contain the name of the ACID:

```
TSS LIST(USER01) KEYRING(ALL)
```

Generate a Certificate Request

You can send a request to a certificate authority to verify the validity of a digital certificate. If CA Top Secret generated the certificate, the request is imported to CA Top Secret just as if the certificate authority was another company.

The request contains the subject's distinguished name and public key and is signed with the private key associated with the specified certificate. A PKCS#10 base64-encoded request is generated and written to data set. The GENREQ DCDSN must not be defined-the output DCDSN cannot be allocated or cataloged, this happens when you use the GENREQ command. The data set can be used as the DCDSN in a TSS GENCERT command.

The GENREQ command generates comments at the beginning of the certificate. Delete the comments if the application accepting the certificate does not support comments.

The syntax for the GENREQ command requires the DCDSN and that you identify the certificate using DIGICERT or LABLCERT (or both).

To generate a certificate request, enter the command:

```
TSS GENREQ(acid|CERTAUTH|CERTSITE) DCDSN(output data set name)
                [DIGICERT(name)]|[LABLCERT('label name')]
```

ACID

A user ACID .

CERTAUTH

Is an ACID in which your installation can maintain certificates that were generated by a third party certificate authority (CA). This ACID is pre-defined. You cannot add a KEYRING to this ACID.

CERTSITE

Is an ACID in which your installation can maintain site-generated certificates. This ACID is pre-defined. You cannot add a KEYRING to this ACID.

DCDSN(output-data-set-name)

The data set will be allocated and cataloged, and will contain the output data set from the genreq'ed digital certificate. The data set name will conform to MVS standards.
Range: Up to 44 characters

DIGICERT

Specifies a case sensitive ID that identifies the certificate with the user ACID.
Range: 1 to 8 characters

LABLCERT

Specifies an optional and case-sensitive label to be associated with the certificate being added to the user. Spaces are allowed if you use single quotes. This label is used as a handle instead of the serial number and issuer's distinguished name, and must be unique for the individual user. If a label is not specified, the label field defaults to the value specified within the DIGICERT keyword.
Range: Up to 32 characters.

Example 1: GENREQ command

This example generates a certificate request:

```
TSS GENREQ(user1) DIGICERT(cert0001)
                  DCDSN(USER3.CERT.DATA)
                  LABLCERT('REQUEST 3')
```

Change a User's Certificate

You can update a user's certificate. If the certificate has a connection to a user key ring, the certificate is replaced and all key ring connections continue with the new certificate. This lets you update a user's certificate without the need to reconnect the certificate to key rings.

To identify the digital certificate to update, use:

- DIGICERT
- LABLCERT
- Both SERIALNUM and ISSUERDN

To change a user's certificate, enter the command:

```
TSS REPLACE(acid|CERTAUTH|CERTSITE) DCDSN(dsname)
             [DIGICERT(name)]
             [LABLCERT(label name)]
             [SERIALNUM(serial number)]
             [ISSUERDN(issuer's dist' name)]
```

SERIALNUM

Specifies the certificate's serial number.

ISSUERDN

Specifies the certificate issuer's distinguished name.

Example: REPLACE command

This example changes USER01's certificate:

```
TSS REPLACE(USER01) DIGICERT(DIGI0001)
                    DCDSN(USER1.CERT.DATA)
```

Certificate Replacement (Renewal)

As part of the TSS REPLACE command processing, a certificate can be replaced without being deleted and reinserted. To replace an existing certificate, make sure that one of the following three cases is satisfied:

- The certificate being added is a duplicate of the existing certificate (it has the same serial number and issuer's distinguished name) and the labels and record keys of both certificates are the same.
- The certificate being added is not a duplicate of the existing certificate, has the same subject's distinguished name, issuer's distinguished name, and public key as the existing certificate, the end date and time on the certificate being added is later than on that of the existing certificate, the existing certificate is not expired, and the record keys of both certificates are the same.
- The certificate being added is not a duplicate of the existing certificate, has the same public key as the existing certificate, there is a private key associated with the existing certificate in the database, the existing certificate is NOT expired, and the record keys of both certificates are the same.

Change a Certificate's Trust Status

The status of the certificate is specified with the TRUST|NOTRUST|HITRUST keyword.

HITRUST

Specifies that the certificate is both highly trusted and trusted. Certificate usage applying to trusted certificates also applies to highly trusted certificates. Only CA certificates (CERTAUTH) can be highly trusted.

TRUST

Specifies that the certificate is valid for the user, site, or CA and the private key is not compromised. On a:

- USER certificate—TRUST indicates that the certificate can be used to authenticate an ACID.
- CERTSITE certificate—TRUST indicates that the certificate can be used without authenticating it.
- CERTAUTH certificate—TRUST indicates that the certificate can be used to authenticate other certificates.

NOTRUST

Indicates that the certificate is not trusted.

The trust status is set to the CA's trust status if the:

- Certificate’s signature can be verified
- Certificate has not expired
- Certificate’s validity dates fall within the range of the signing certificate

The default trust status for self-signed certificates is TRUST.

The trust status is set to NOTRUST if the certificate being added or generated:

- Has expired
- Has an invalid validity date range
- Cannot be verified

The trust status of the new certificate is set to TRUST if the trust status coming from the signing certificate is HITRUST.

To identify the digital certificate to update use:

- DIGICERT
- LABLCERT
- Both SERIALNUM and ISSUERDN

To change the status of a certificate, enter the command:

```
TSS REPLACE(acid|CERTAUTH|CERTSITE) [DIGICERT(name)]
                                    [LABLCERT(label name)]
                                    [SERIALNUM(serial number)]
                                    [ISSUERDN(issuer's dist name)]
                                    TRUST|NOTRUST|HITRUST
```

Example: replacing status

This example changes a certificates status:

```
TSS REPLACE(user1) DIGICERT(cert0001)
                   NOTRUST
```

Signed Certificates with Unspecified Trust Status

The following applies when the trust status is not specified and the certificate is signed by another certificate:

- If all of the following conditions are met, the status is set to TRUST:
  - The signing certificate can be located in the database.
  - The signing certificate is a Certification Authority (CERTAUTH).
  - The signing certificate has HITRUST
  - The signing certificate has not expired.
  - The validity dates of the certificate being added fall within the range of the signing certificate's validity dates.
- If any of the above conditions are not met, the certificate is added as NOTRUST and a message is issued stating why.

Trust Status with PKCS 7 and PKCS 12 Certificates

On an add of PKCS 7 certificate packages, if there is more than one certificate in the package, the 2nd certificate through the last certificate are considered to be CA certificates.

On an add of PKCS 12 certificate packages, any certificate that does not have a “local key id” is considered to be a CA certificate. The “local key id" is a string that allows keys and certificates to be matched (same id means matched).

The CA certificates are sorted to determine the hierarchy. They are then added under the CERTAUTH ACID from the top CA to the lowest CA so that each certificate in the package can be verified using it's previously added signing certificate. Then the end-entity certificate is added.

The CA certificates added have a record ACID and label in the format CERTAUTH.AUTOnnnn, where the nnnn is a number between 0 and 9999. The highest level CA certificate will not necessarily have an AUTOnnnn number less that the other CA certificates being added.

When CA certificates are added from a PKCS 7 or PKCS 12 chain, the following rules apply:

- If the certificate has already been added and has HITRUST status, the certificate retains the HITRUST status.
- The trust status specified on the command applies to the top level CA certificate. This sets up the TRUST status so that subsequent adds of the other CA certificates and end-entity certificate can inherit the status from it's signing certificate. HITRUST status is ignored if the record is for an ACID other then CERTAUTH.
- When no trust status is specified for all lower CA's in the certificate chain:
  - If the certificate has one or more of the following inconsistencies, the certificate is added with NOTRUST status:
    - The certificate is expired
    - The validity period does not fall within the signer's validity period
    - The issuer of the certificate is missing from the package and is not already in the CA Top Secret database
    - The certificate has an unknown signature algorithm
  - If no inconsistencies are detected, the certificate is added and inherits the trust status of the signing certificate
- HITRUST is inherited from the parent only if the target ACID on the add command is CERTAUTH. In all other cases, the trust status changes to TRUST.
- If an error occurs during an add from a PKCS 7 or PKCS 12 certificate package, there is no back-out processing. Certificates already added are not removed.
- If the above rules conflict, the first one that matches applies.

Change a Certificate's Label

The label for a certificate can be changed. The certificate label being updated must be identified using DIGICERT or SERIALNUM and ISSUERDN.

To change a certificates label, enter the command:

```
TSS REPLACE(acid|CERTAUTH|CERTSITE) [DIGICERT(name)]
                                    [SERIALNUM(serial number)]
                                    [ISSUERDN(issuer's dist name)]
                                    LABLCERT(label name)
```

Example: replacing a label

This example replaces a certificates label:

```
TSS REPLACE(USER01) DIGICERT(DIGI0001)
                    LABLCERT('label for digicert 002')
```

Remove a Certificate from a User

Use the REMOVE command to remove a certificate from a user. Issuing the command also removes any key ring connections.

If the GENREQ command has been issued against a certificate (when creating a certificate based on an original certificate), the original certificate cannot be removed unless you specify the FORCE operand or the new certificate has replaced the original certificate.

You can use the following keywords to identify the digital certificate:

- DIGICERT
- LABLCERT
- Both SERIALNUM and ISSUERDN

To remove a certificate from a user, enter the following command:

```
TSS REMOVE(acid|CERTAUTH|CERTSITE)      [DIGICERT(eight_byte_name)]
                                        [LABLCERT(label_name)]
                                        [SERIALNUM(serial_number)]
                                        [ISSUERDN(issuer_distinguished_name)]
                                        FORCE
```

***acid*|CERTAUTH|CERTSITE**

Provides one of the following functions:

- Specifies the ACID of the user associated with the certificate
- Identifies the certificate as a certificate-authority certificate (CERTAUTH)
- Identifies the certificate as a site certificate (CERTSITE)

DIGICERT

Specifies a name to identify the digital certificate.

LABLCERT

Specifies a label associated with the certificate.

SERIALNUM

Specifies the serial number of the certificate.

ISSUERDN

Specifies the certification authority's distinguished name as extracted from the certificate.

FORCE

Forces the removal of the certificate from the user when the certificate has been GENREQ'd. This action should be done only in the following situations:

- You specified GENREQ when REKEY was intended.
- You specified REKEY when GENREQ was intended.

Example: Remove a Certificate

This example removes a certificate:

```
TSS REMOVE(USER01) DIGICERT(DIGI0001)
```

Renew an Existing Certificate

Valid on z/OS.

The RENEW command function renews a digital certificate. Use the RENEW command if a certificate is expiring and you want to continue to use the certificate. The RENEW command copies the information from an existing certificate and applies it to a certificate. This command can only renew certificates created by CA Top Secret, it cannot renew certificates from external certificate authorities. Certificates from external certificate authorities must be renewed manually using the PKCS 10 data set.

Notes:

- Because the DIGICERT keyword indicates the name used in the digital certificate, specify a DIGICERT name as part of all RENEW functions.
- You can enter additional keywords to modify the renewing certificate.
- Use the NADATE keyword to specify an expiration date. If no date is specified, the certificate expires one year from the date that the certificate is renewed.
- The keywords ICSF, PCICC, and LABLPKDS are only valid if the original certificate is NON-ICSF.
- Before you renew a certificate, we recommend as a best practice that you export the certificate to a data set to save it, either using the RENEW command or manually.

Administrators must have:

- ACID(MAINTAIN) and MISC4(CERTGEN) for users within their scope
- MISC4(CERTSITE) for CERTSITE ACID
- MISC4(CERTAUTH) for CERTAUTH ACID

This command function has the following format:

```
TSS RENEW {(CERTAUTH|CERTSITE|acid)}
    DIGICERT(8-byte-name)
    [SUBJECTN('CN="common-name" T="title" OU="org-unit-name1,org-unit-name2"
    O="organizational-name" L="locality" ST="state-or-province" C="2–digit–only
    country code"')]
    [NBDATE(mm/dd/yy) NBTIME(hh:mm:ss)]
    [NADATE(mm/dd/yy) NATIME(hh:mm:ss)]
    [LABLCERT(label name)]
    [ICSF|PCICC]
    [SIGNWITH(acid,digicert)]
    [KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN CERTSIGN)]
    [ALTNAME('IP=numeric-IP-address DOMAIN=internet-domain-name
    EMAIL=email-address URI=universal-resource-identifier')]
    [LABLPKDS]
```

CERTAUTH|CERTSITE|acid

Specifies the acid that will use the certificate.

CERTAUTH

Specifies the certificate as a certificate-authority certificate.

CERTSITE

Specifies the certificate as a site certificate.

acid

Specifies the user associated with the certificate.

DIGICERT(8-byte-name)

Specifies the keyword that identifies the digital certificate being renewed.

SUBJECTN('CN="common-name" **T=**"title" **OU=**"org-unit-name1,org-unit-name2" **O=**"organizational-name" **L=**"locality" **ST=**"state-or-province" **C=**"2—digit—only country code"**')**

(Optional) Specifies the distinguished name of the ACID. When you specify multiple parameters for SUBJECTN, surround the parameter list with single quotation marks. You can specify multiple values for OU=.

Default: The name field of the ACID

NBDATE/NBTIME Format(*mm/dd/yy*) Time(*hh:mm:ss*)

Indicates the date and time that the certificate becomes active. If no expire date is specified, the active year specified must be before 2048, because the expire date defaults to the active day and time plus one year.

Range: 1950 to 2049

Time Default: 000000

Date Default: Current day and time

NADATE/NATIME Format(*mm/dd/yy*) Time(*hh:mm:ss*)

(Optional) Indicates the date and time that the certificate expires.

Range: 1950 to 2049

Time Default: 000000

Date Default: The active day and time plus one year.

LABLCERT(*label-name*)

(Optional) Defines the label name of the certificate being renewed.

ICSF

(Optional) Indicates that the generated private key is placed in ICSF. If ICSF, PCICC, or LABLPKDS is not specified with ADD, the key is stored in the security file as a non-ICSF key. If the DSN parameter was also specified and an existing certificate is replaced, the existing certificate is also placed in ICSF. If ICSF is not active and configured for PKA operations, an error message is displayed when attempting to insert or use the private key.

PCICC

(Optional) Specifies that the key pair is generated using the PCI Cryptographic Coprocessor and that the private key is stored in ICSF. When PCICC is not specified, the key pair is generated using software. PCICC cannot be used with the DSA, DSN, or ICSF parameters.

If a PCI cryptographic coprocessor is not present or operational or if ICSF is not active or configured for PKA operations, an error message is displayed and processing terminates. If ICSF, PCICC, or LABLPKDS is not specified, the key pair is generated using software and stored in the security file as a non-ICSF key.

SIGNWITH(acid,digicert)

(Optional) Specifies the digital certificate signing the certificate. If not specified, the certificate is signed with the private key of the certificate being generated, creating a self-signed certificate.

KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN CERTSIGN)

(Optional) Specifies the appropriate values for the KeyUsage certificate extension. If the KEYUSAGE data contains more than one value, place single quotation marks around the data.

Example: KEYUSAGE('HANDSHAKE DATAENCRYPT')

HANDSHAKE

(Optional) Facilitates identification and key exchange during security handshakes, such as SSL, which set the digitalSignature and keyEncipherment indicators. When the key pair is generated using the DSA algorithm, only the digitalSignature bit is set because the keys cannot be used for encryption.

DATAENCRYPT

(Optional) Encrypts data, which sets the dataEncipherment indicator. When the key pair is generated using the DSA algorithm, you cannot use the DATAENCRYPT keyword in the Keyusage parameter.

DOCSIGN

(Optional) Specifies a legally binding signature, which sets the nonRepudiation indicator.

CERTSIGN

(Optional) Specifies a signature for other digital certificates and CRLs, which sets the keyCertSign and cRLSign indicators.

ALTNAME ('IP=numeric-IP-address **DOMAIN=**internet-domain-name **EMAIL=**email-address **URI=**universal-resource-identifier**')**

(Optional) Specifies the appropriate values for the SubjectAltname extension. When you specify multiple parameters for ALTNAME, surround the parameter list with single quotation marks. Separate multiple parameters by a space.

Example: ALTNAME('IP=200.100.10.1 EMAIL=my.email@test.net')

LABLPKDS

(Optional) Specifies the PKDS label of the record created in the ICSF Public Key Data Set (PKDS). This field is used with the ICSF, PCICC, NISTECC, and BPECC keywords. If LABLPKDS is specified without ICSF or PCICC, the key is generated by the hardware and saved in CRT format in the ICSF PKDS. If NISTECC or BPECC is specified, the key is an ECC key; otherwise, the key is an RSA key.

Note: The PKDS label must conform to ICSF label syntax rules. The first character must be alphabetic or national.

To take the value from the LABLCERT keyword, include a LABLCERT specification in your syntax and specify the following syntax for LABLPKDS:

```
LABLPKDS(*)
```

Valid characters: Alphanumeric, national (@,#,$), or period(.).

Range: Up to 64 characters

Example: RENEW function

This example creates new certificate Locca4 for the existing certificate CERTAUTH.

```
TSS RENEW(CERTAUTH) DIGICERT(Locca4) NADATE(12/31/11)
```

When to Use RENEW and REKEY

Use the RENEW command function if a certificate is expiring and you want to continue to use the certificate. You can also use RENEW to alter the attributes of the certificate at the time of renewal. For example, to:

- Alter the validity period of the certificate
- Alter the certificate's distinguished name
- Add or remove an ALTNAME value

Use the REKEY command function to change the public/private key pair, change the size or type of the keys, or change from RSA to DSA. An appropriate time to use REKEY is when the existing public/private key pair may be compromised, or when a particular key size no longer meets company standards.

For more information on the REKEY command function, see the *Command Functions Guide*.

Determine Certificate Associations

You can check to see who has a certificate in a specified data set. The CHKCERT command determines whether the digital certificate in the specified data set has been added to the CA Top Secret security file and associated with an ACID.

Note: The CHKCERT command is the only way to display the ALTNAME parameter for a certificate. Only one domain name is displayed, even though multiple domain names exist.

To determine the association details of a certificate, enter the following command:

```
TSS CHKCERT DCDSN(request_dataset_name)
            PKCSPASS(pksc12_password)
            DUMP
            CHAIN
```

DCDSN(*request_dataset_name*)

Specifies the name of an optional data set that contains the PKCS#12 certificate request data. The request data set name can be the output from a TSS GENREQ command. The request data contains the user's generated public key and X.509 distinguished name. The request data must be signed, DER-encoded, and then Base64-encoded according to PKCS#12 standard.

PKCSPASS

Specifies a case-sensitive PKCS-password that can contain blanks.

Range: Up to 255 characters

DUMP

(Optional) Displays the contents of the user certificate in hexadecimal format.

CHAIN

Displays information for each certificate in the chain of the input data set and displays the following summary Information as applicable:

- Number of certificates in the chain
- Indication of whether the chain is complete or incomplete
- Indication of whether the chain contains expired or non-trusted certificates.
- Indication of whether any certificate in the data set is not present in the CA Top Secret database.

Important! Passwords associated with PKCS#12 certificates are not viewable. It is the CA Top Secret administrator's responsibility to keep track of the PKCS#12 password that is assigned to the digital certificate.

Example: Display Certificate Associations

This example uses the DCDSN keyword to specify a certificate package:

```
TSS CHKCERT DCDSN(reipa02.user2.cert2)
```

The product reviews each certificate in the package. If a certificate is in the database, the product lists the user and ID to which the certificate is defined.

Export Certificates to Data Sets

You can export a certificate from a CA Top Secret security file to a new data set. The certificate can be identified by its DIGICERT name or by its label.

If the certificate's private key resides in an ICSF storage facility and the format of PKCS12DER or PKCS12B64 is specified in the TSS EXPORT command, the command is rejected. You cannot export a digital certificate with ICSF.

We recommended that you export a certificate to a dataset in order to save it before renewing, both when using the RENEW command and when renewing by manual process.

To export a certificate to a data set, enter the command:

```
TSS EXPORT(acid|CERTAUTH|CERTSITE)
                [DIGICERT(name)
                [LABLCERT(labelname)]
                [DCDSN(output-data set name)]
                [FORMAT(format type)]
                [PKCSPASS(PKCS#12 password)]
```

ACID

A user ACID.

CERTAUTH

Is an ACID in which your installation can maintain certificates that were generated by a third party certificate authority (CA). This ACID is pre-defined. You *cannot* add a KEYRING to this ACID.

CERTSITE

Is an ACID in which your installation can maintain site-generated certificates. This ACID is pre-defined. You *cannot* add a KEYRING to this ACID.

DIGICERT

Specifies a case sensitive character ID that identifies the certificate with the user ACID. The DIGICERT must be entered as part of all GENCERT commands since this keyword indicates the name to be used in the digital certificate.

Range: 1 to 8

DCDSN(output-data-set-name)

The data set will be allocated and cataloged, and will contain the output from the exported digital certificate. The data set must conform to the MVS standards.
Range: Up to 44 characters

FORMAT

The following operands can be used with the FORMAT keyword:

CERTB64

(Default) Indicates Base64 encoded certificates.

CERTDER

Indicates DER encoded X.509 certificates..

PKCS7B64

Specifies a B64 encoded PKCS#7 package.

PKCS7DER

Specifies a DER encoded PKCS#7 package.

PKCS12B64

Indicates DER encoded (then Base64 encoded) PKCS#12 package.

PKCS12DER

Indicates DER encoded PKCS#12 package.

You only get a private key if specifying with 'PKCS" format.

LABLCERT

Specifies an optional and case-sensitive label to be associated with the certificate being added to the user. Spaces are allowed if you use single quotes. This label is used as a handle instead of the serial number and issuer's distinguished name, and must be unique for the individual user. If a label is not specified, the label field will default to the value specified within the DIGICERT keyword.
Range: Up to 32 characters.

PKCSPASS

The PKCS-password is case sensitive and can contain blanks.
Range: Up to 255 characters

Example: Export a certificate

This example exports a certificate:

```
TSS EXPORT(USER01) DIGICERT(DIGI0001)
                   DCDSN(USER3.CERT.DATA)
                   FORMAT(CERTDER)
```

Export a Certificate with Private and Public Keys

To export both the public and private keys with the certificate:

- Use the TSS EXPORT command with:
  - One of the PKCSxxxx formats.(the default format, CERTB64 exports the public key only)
  - A password specified with the PKCSPASS keyword
- When adding an exported certificate use the PKCSPASS keyword to specify the password used to protect the certificate during the TSS EXPORT

Example: exporting a certificate with private and public keys

This example exports both keys with the certificate:

```
TSS EXPORT(TESTXXX) DIGICERT(TESTSYS)
                    DCDSN('TESTSYS.CERT.PKCS')
                    FORMAT(PKCS12DER)
                    PKCSPASS(user_specified_password)

TSS ADD(acid) DIGICERT(TESTSYS)
              LABELCERT(certificate label)
              DCDSN('TESTSYS.CERT.PKCS')
              TRUST PKCSPASS(user_specified_password)
```

REKEY Function—Create Certificate from Existing Certificate

Valid on z/OS.

Use the REKEY command function to create a new certificate from an existing certificate with a new public/private key pair. The REKEY command is the first step of a rekey-rollover process to retire the use of an existing private key.

The REKEY command copies the subject's distinguished name, key usage and subject alternate name from the existing certificate. The new certificate is self-signed and saved under the same logonid or CERTAUTH or CERTSITE.

If the new certificate needs to be signed by a third party CA or CA Top Secret, issue a TSS GENREQ command to copy the new certificate to a dataset then FTP the new certificate to the third party CA or input to the TSS GENCERT so it may be signed with CA Top Secret. Do this prior to the TSS ROLLOVER command.

Specify a DIGICERT name as part of all REKEY functions since the DIGICERT keyword indicates the name used in the digital certificate.

Administrators must have:

- ACID(MAINTAIN) and MISC4(CERTGEN) for users within their scope
- MISC4(CERTSITE) for CERTSITE ACID
- MISC4(CERTAUTH) for CERTAUTH ACID
- Administrators without the above authorities can issue the REKEY command if they have:
  - UPDATE access to TSSCMD.CERTUSER.REKEY in the CASECAUT resource class when the certificate is associated with a user ACID
  - UPDATE access to TSSCMD.CERTSITE.REKEY in the CASECAUT resource class when referencinging a site certificate
  - UPDATE access to TSSCMD.CERTAUTH.REKEY in the CASECAUT resource class when referencinging a certificate-authority certificate

This command function has the following format:

```
TSS REKEY {acid|CERTAUTH|CERTSITE}
          [DIGICERT(existing–certificate–id)]
          [NEWDIGIC(new–certificate–id)]
          [NEWLABLC(new–certificate–label)]
          [KEYSIZE(nnnn)]
          [ICSF|PCICC|NISTECC|BPECC]
          [NBDATE({not–before–date} NBTIME(not–before–time)]
          [NADATE(not–after=date|) NATIME(not–after–time)]
          [LABLPKDS]
```

DIGICERT(*id*)

(Mandatory) Specifies a case sensitive character ID that identifies existing certificate.

Range: 1 to 8 characters

NEWDIGIC(*id*)

(Mandatory) Specifies a case sensitive character ID of the new certificate.

NEWLABLC(*label*)

(Optional) Specifies the new certificate's a character label. The label can contain blanks and mixed case characters. The new label must be unique to the logonid with which the new certificate is associated. If a label is not specified, the label field defaults to the upper case version of the ACID.

Note: For every one apostrophe desired in the Label value, two consecutive apostrophes must be specified. For example, the Label value, Frank's Certificate, should be specified as, Frank”s Certificate. If a single apostrophe is specified in the Label value, the value is considered invalid.

Range: 1 to 32 characters

KEYSIZE

- Specifies the size of the private encryption key in decimal bits.
- **Limits:**
  - Non-NISTECC and non-BPECC: any numeric value from 512 to 4096.
  - For NISTECC, one of the following: 192, 224, 256, 384, 521.
  - For BPECC, one of the following: 160, 192, 224, 256, 320, 384, 512.
- **Default:**
  - Non-NISTECC and non-BPECC:1024.
  - NISTECC and BPECC: 192.

The maximum key size is dependent on the private key type.

Private key type maximum key sizes are:

- BPECC key—512 bits
- NISTECC key—521 bits
- ICSF RSA key—1024 bits
- DSA key —2048 bits
- Non-ICSF RSA key—4096 bits
- PCI-class cryptographic coprocessor RSA key—4096 bits

Shorter ECC keys have key strengths comparable to longer RSA keys. The following table displays the comparable strength of each key type:

RSA Key Size (in bits)	NISTECC Key Size (in bits)	BPECC Key Size (in bits)
1024	192	160 or 192
2048	224	224
3072	256	256 or 320
7680	384	384
15360	521	512

Currently, the standard key sizes for RSA keys are as follows:

- 512—Specifies a low-strength key
- 1024—Specifies a medium-strength key
- 2048—Specifies a high-strength key
- 4096—Specifies a very high-strength key

ICSF

(Optional) Indicates that the generated private key is placed in ICSF. If ICSF, PCICC, or LABLPKDS is not specified with ADD, the key is stored in the security file as a non-ICSF key. If the DSN parameter was also specified and an existing certificate is replaced, the existing certificate is also placed in ICSF. If ICSF is not active and configured for PKA operations, an error message is displayed when attempting to insert or use the private key.

PCICC

(Optional) Specifies that the key pair should be generated using the PCI Cryptographic Coprocessor and that the private key should be stored in ICSF. When PCICC is not specified, the key pair is generated using software. PCICC cannot be used with the DSA, DSN, or ICSF parameters. If a PCI cryptographic coprocessor is not present or operational, or if ICSF is not active or configured for PKA operations, an error message is displayed and processing will terminate.

NISTECC

(Optional) Specifies the key pair should be generated using National Institute of Standards and Technology (NIST) algorithm instead of the RSA algorithm. This parameter cannot be used with the ICSF, DSA, or BPECC parameters.

BPECC

(Optional) Specifies to generate the key pair using the brainpool ECC algorithm instead of the RSA algorithm. This parameter cannot be used with the ICSF, DSA, or NISTECC parameters.

Notes:

- If you do not specify ICSF, PCICC, NISTECC, or BPECC, the key pair is generated using software and stored in the CA Top Secret database.
- REKEY processing prevents the downgrade from an ICSF/PCICC private key to a private key stored in the security product database.
- If you do not specify ICSF, PCICC, NISTECC, or BPECC, REKEY will use what was used on the original certificate.
- REKEYing from a NISTECC or BPECC certificate to a non-ECC certificate (and conversely) is not allowed.
- You do not need to specify anything to go from NISTECC to NISTECC or from BPECC to BPECC.
- Specify NISTECC to go from BPECC to NISTECC (and conversely).

NBDATE/NBTIME Format(*mm/dd/yy*) Time(*hh:mm:ss*)

Indicates the date and time that the certificate becomes active. If no expire date is specified, the active year specified must be before 2048, because the expire date defaults to the active day and time plus one year.

Range: 1950 to 2049

Time Default: 000000

Date Default: Current day and time

NADATE/NATIME Format(*mm/dd/yy*) Time(*hh:mm:ss*)

(Optional) Indicates the date and time that the certificate expires.

Range: 1950 to 2049

Time Default: 000000

Date Default: The active day and time plus one year.

LABLPKDS

(Optional) Specifies the PKDS label of the record created in the ICSF Public Key Data Set (PKDS). This field is used with the ICSF, PCICC, NISTECC, and BPECC keywords. If LABLPKDS is specified without ICSF or PCICC, the key will be generated by the hardware and saved in CRT format in the ICSF PKDS. If NISTECC or BPECC is specified, the key will be an ECC key. Otherwise, the key that is generated will be an RSA key.

Specify a value of (*) to take the value from the LABLCERT keyword. In this case, LABLCERT must be specified along side LABLPKDS(*). If LABLPKDS(*) is specified without the LABLCERT keyword, an error message is displayed.

The PKDS label must conform to ICSF label syntax rules. The first character must be alphabetic or national.

Valid characters: Alphanumeric, national (@,#,$) or period(.).

Range: Up to 64 characters

Example: REKEY function

This example creates the new certificate Locca4 for the existing certificate CERTAUTH.

```
TSS REKEY(CERTAUTH) DIGICERT(Locca4)
                    NEWDIGIC(locca5)
                    NADATE(12/31/08)
```

ROLLOVER Function—Specify Original Certificate

Valid on z/OS.

Use the ROLLOVER command function to specify the original certificate superseded by the new certificate. The ROLLOVER sub-command is the final step in the REKEY command, rollover process.

The ROLLOVER command function:

- Deletes the private key of the original certificate so that it can no longer be used to sign or encrypt
- Replaces the original certificate with the new certificate in every key ring that the old certificate is connected to
- Copies the serial number base from the original certificate to the new certificate

When the rollover is complete, the new certificate is used as if it were the original certificate. The original certificate is still available to verify signatures and decrypt data, but can no longer be used to sign or encrypt.

Specify a DIGICERT and NEWDIGIC names as part of all ROLLOVER functions since the keywords indicates the names used in the digital certificate ROLLOVER command.

Administrators must have:

- ACID(MAINTAIN) and MISC4(CERTGEN) authority for users within their scope
- MISC4(CERTSITE) authority for CERTSITE ACID
- MISC4(CERTAUTH) authority for CERTAUTH ACID
- Administrators without the above authorities can issue the ROLLOVER command if they have:
  - UPDATE access to TSSCMD.CERTUSER.ROLLOVER in the CASECAUT resource class when the certificate is associated with a user ACID
  - UPDATE access to TSSCMD.CERTSITE.ROLLOVER in the CASECAUT resource class when referencing a site certificate
  - UPDATE access to TSSCMD.CERTAUTH.ROLLOVER in the CASECAUT resource class when referencing a certificate-authority certificate

This command function has the following format:

```
TSS ROllOVER {acid|CERTAUTH|CERTSITE|}
             [DIGICERT(old–certificate–id)]
             [NEWDIGIC(new–certificate–id)]
             [Forcer]
```

acid

Designates the user ACID associated with the certificate.

CERTAUTH

Designates the certificate as a certificate-authority certificate.

CERTSITE

Designates the certificate as a site certificate.

DIGICERT(id)

Specifies a case-sensitive character ID (original certificate) that identifies the certificate with the user ACID.
(Mandatory with ROLLOVER keyword.)

Range: 1 to 8 characters

NEWDIGIC(id)

Specifies a case-sensitive character ID that identifies the new certificate.
(Mandatory with ROLLOVER keyword.)

Range: 1 to 8 characters

FORCER

Specifies that CA Top Secret should bypass the following checks and perform the rollover unconditionally.

- The values of DIGICERT and NEWDIGIC must be different
- The certificates identified by DIGICERT and NEWDIGIC must both have private keys associated with them
- The certificate identified by NEWDIGIC must have never been the target of a previously issued ROLLOVER sub-command and never used to sign other certificates

When the FORCER keyword is specified, the previous three checks are not performed.

Note: The ROLLOVER sub-command has a degenerative feature where the private key of the certificate is deleted if both DIGICERT and NEWDIGIC are the same and the FORCER keyword is also used.

Example: ROLLOVER function

This example completes the re-keying of the TEN certificate.

```
TSS ROLLOVER(CERTSITE) DIGICERT(NINE)
                       NEWDIGIC(TEN)
                       FORCER
```

Replace an Expired Certificate

Use the REKEY and ROLLOVER commands to replace an expired certificate.

The REKEY command:

- Creates a new certificate from an existing certificate with a new public/private key pair
- Copies the subject's distinguished name, key usage and subject alternate name from the existing certificate

The ROLLOVER command:

- Specifies the original certificate to be superceded by the new certificate
- Deletes the private key of the original certificate so that it can no longer be used to sign or encrypt
- Replaces the original certificate with the new certificate in every key ring the old certificate is connected to
- Copies the serial number base from the original certificate to the new certificate

The new certificate:

- Is self-signed
- Is saved under the same logonid or CERTAUTH or CERTSITE
- Is used as if it were the original certificate

The original certificate can still verify signatures and decrypt data, but cannot sign or encrypt.

Example: replacing an expired certificate

In this example, the ACID 'CERTSITE' is the owner of certificate JOECERT1.

DIGICERT(JOECERT1) with a LABLCERT(JOECERT1) has been given to 1000 keyrings. Now, JOECERT1 has expired and needs to be replaced with a new Digital Certificate.

1. Enter the command:

   ```
   TSS REKEY(CERTSITE) DIGICERT(JOECERT1) NEWDIGIC(JOECERT2)
   ```

   A new certificate called JOECERT2 based on the expired certificate JOECERT1 is created.

2. Enter the command:

   ```
   TSS GENREQ(CERTSITE) DIGICERT(JOECERT2) DCDSN(JOECERT2.CERT.UNSIGNED)
   ```

   JOECERT2 is copied to a data set.

3. FTP the certificate to be signed by the third-party Certificate Authority.
4. Enter the command:

   ```
   TSS REP(CERTSITE) DIGI(JOECERT1) LABLCERT('JOECERT1 OLD')
   ```

   The LABLCERT of JOECERT1 is renamed.
5. Enter the command:

   ```
   TSS ADD(CERTSITE) DIGICERT(JOECERT3)
                     DCDSN(JOECERT2.CERT.SIGNED)
                     TRUST LABLCERT('JOECERT1')
   ```

   The signed certificate is added to CA Top Secret under a new DIGICERT name called JOECERT3 and a LABLCERT of JOECERT1.
6. Enter the command:

   ```
   TSS ROLLOVER(CERTSITE) DIGICERT(JOECERT1) NEWDIGIC(JOECERT3)
   ```

   The new JOECERT3 certificate is propagated to the 1000 keyrings.

Add a Key Ring to an ACID

To add a key ring to an ACID, enter the command:

```
TSS ADD(acid) KEYRING(key ring)
              [LABLRING(ring name)]
```

KEYRING

Specifies the key ring being added to the user's ACID. An individual ACID can be a member of more than one key ring.

Range: Up to 8 characters

LABLRING

Specifies the label to be associated with the key ring being added to the user. This label is used as an identifier of the digital certificate code and must be unique for the key ring.

Range: Up to 237 characters

Add a Certificate to a Key Ring

You can add digital certificates issued by a certificate authority to one user to another user's key ring. This allows the administrator to further define the access that a user has to certain resources. Before a digital certificate can be added to a key ring, it must have been added to the owner's ACID record through the TSS ADD DIGICERT command.

To add a keyring to a user, enter the command:

```
TSS ADD(acid) KEYRING(ring name)
              LABLRING(name)
```

To add a certificate to a key ring, enter the command:

```
TSS ADD(acid) KEYRING(ring name)
              [LABLRING(name)]
              [RINGDATA(acid,digicert)]
              [RINGDATA(CERTAUTH,digicert)]
              [RINGDATA(CERTSITE,digicert)]
              [DEFAULT]
              [USAGE(PERSONAL|CERTAUTH|CERTSITE)]
```

KEYRING

The ring name is unique within the user, the name you specify identifies the key ring for a user.

Range: Up to 8 characters

LABLRING

Provides the ability add a label name to the key ring; can be used as a key to locate a certificate key ring. If not specified, the KEYRING name is automatically added to the LABLRING.

Range: Up to 237 characters

RINGDATA

Specifies the ACID and certificate label name (as specified by DIGICERT) of the certificate being added to the user.

DEFAULT

(Optional) Specifies that the certificate is the default certificate for the key ring.Only one certificate within the key ring can be the default. If a default already exists, its DEFAULT status is removed, and the specified certificate becomes the default certificate.

USAGE

Specifies how this certificate is used with the specified key ring. The default usage is the same as the certificate being connected.

PERSONAL

Demotes a certificate to ensure that it is not used as a certificate authority in this key ring.

CERTAUTH

Promotes an ordinary user certificate to that of a certificate authority within this key ring.

CERTSITE

Promotes an ordinary user certificate to that of a site certificate.

Remove a Key Ring from an ACID

Removing a key ring also removes all key ring cross references from all ACIDs that have certificates attached to the key ring being removed.

To remove a key ring, enter the command:

```
TSS REMOVE(acid) KEYRING(name)|LABLRING(name)
```

Extract Certificates from Key Rings

Authorized applications, such as servers HTTP, TN3270, CICS, or LDAP, invoke the R_Datalib callable service (IRRSDL00) to retrieve certificates and private keys from a key ring, and manage serial numbers for certain certificates.

CA Top Secret supports the R_Datalib functions using its Keyring support. Authorize these accesses to IRRSDL00 functions by administering CA Top Secret resource class (IBMFAC) facility permissions for the IRR.DIGTCERT.*function*. Where *function* could be LISTRING, LIST, or GENCERT.

Example: extract a certificate from a key ring

This example extracts a user certificate from a key ring, you require access to IBMFAC function LISTRING:

```
TSS ADD(dept) IBMFAC(IRR.DIGTCERT)

TSS PER(acid) IBMFAC(IRR.DIGTCERT.LISTRING)
              ACCESS(UPDATE)

TSS PER(acid) IBMFAC(IRR.DIGTCERT.LIST)
              ACCESS(UPDATE)

TSS PER(acid) IBMFAC(IRR.DIGTCERT.GENCERT)
              ACCESS(UPDATE,CONTROL)
```

Note: If the certificate user ID is the same as the user ID issuing the R-Datalib call, the required authority is ACCESS (READ). If the user ID is not the same the required authority is ACCESS (UPDATE) or ACCESS (CONTROL).

Extract Certificates from Virtual Key Rings

A virtual key ring (VKR) is a set of certificates for a specific user or server application used to determine the trustworthiness of a client or peer entry. VKRs are requested when the ring name passed to R_datalib is a single asterisk (*). All trusted certificates for the requested user (RTSS_user_ID parameter) are returned.

There is no default certificate. If the requested user is CERTAUTH, all trusted CERTAUTH certificates are returned. If the requested user is CERTSITE, all trusted CERTSITE certificates are returned.

Extract Private Keys

An application can extract the private key from a user certificate if the following conditions are met:

- The caller's user ID is the user ID associated with the certificate
- The certificate is connection to its key ring with the PERSONAL usage option

An application can extract the private key from a CERTAUTH or CERTSITE certificate if the following conditions are met:

- The caller's user ID has at least CONTROL access to the IBMFAC resource IRR.DIGTCERT.GENCERT
- The certificate is connection to its key ring with the PERSONAL usage option

Reconnect Private Keys

When generating self-signed certificates using GENCERT, a public/private key pair is built and stored within the certificate. The private key always remains with the certificate unless it is sent to a third-party as a certificate request. When a GENREQ certificate request is sent to a third-party, the returned certificate will not contain the private key. This happens because private keys are not shipped as part of a certificate request.

To use a third-party certificate, the private key must be re-connected to the certificate. This is accomplished automatically when a TSS ADD command is issued to re-connect the third-party certificate to the same user id that has the (model) certificate. The original, self-signed certificate private key, is connected to the new certificate.

As long as the user ID is the same and the public key within the third-party certificate matches the original certificate, the private key is connected.

Key Ring Information

To list all the ACIDs and their key rings associated with them on a system using VSAM/R9 enter:

```
TSS LIST(SDT) KEYRING(ALL)
```

To list all the ACIDs and their key rings associated with them on a system using VSAM/R12 enter:

```
TSS LIST(ACIDS) KEYRING(ALL)
```

To list the associated Key Ring and LABLRING for a specific ACID enter one of:

```
TSS LIST(USER01) KEYRING(RING0001)
```

```
TSS LIST(USER01) LABLRING(LABELRING0002)
```

The command must contain the name of the Key Ring or LABLRING already associated with the ACID.

Certificate Serial Numbers

An application can invoke the R_Datalib callable service to manage serial numbers for certificates.

An application can increment the “last serial number issued” for a personal (user) certificate if the following conditions are met:

- The caller's user ID is the user ID associated with the certificate
- The caller's user ID has at least READ authority to the IBMFAC resource IRR.DIGTCERT.GENCERT.

```
TSS PER(acid) IBMFAC(IRR.DIGTCERT.GENCERT)
              ACCESS(READ)
```

An application can increment the “last serial number issued” for a CERTAUTH or CERTSITE certificate if the following conditions are met:

- The caller's user ID is the user ID associated with the certificate
- The caller's user ID has at least CONTROL authority to the IBMFAC resource IRR.DIGTCERT.GENCERT

```
TSS PER(acid) IBMFAC(IRR.DIGTCERT.GENCERT)
              ACCESS(CONTROL)
```

Move a Certificate to Another System

You can move a digital certificate from one system to another system that does not share the Security file. You do not have to use the the same ACID or the same digicert name on the new system.

To export a certificate to another system

1. From the system with the certificate, enter the command:

```
TSS EXPORT(acid) DIGICERT(certname)
                 DCDSN(datasetname)
                 PKCSPASS(password)
                 FORMAT(format-type)
```

2. If the DCDSN is not on shared DASD, FTP the certificate to the new system.
3. From the new system, enter the command:

```
TSS ADD(acid) DIGICERT(certname)
              DCDSN(datasetname)
              PKCSPASS(password)
              TRUST
```

Certificate Name Filtering Support

Certificate name filtering (CNF) support allows certificates to be associated with users without having to add each certificate to the CA Top Secret security file. This decreases the amount of storage and the administration needed to support a large number of certificates.

Certificate name filtering allows profiles based on the certificate subject/issuer distinguished name to be used to select the ACID to assign for a particular certificate. Many certificates can be associated with a single ACID. This support provides more granular access control and accountability.

When a certificate name filter is defined, the information is stored in a CERTMAP record in the SDT on the security file. The filter definition specifies the significant portion of the issuer's or subject's distinguished name that is used to associate an ACID with a certificate.

Additional criteria can be specified to identify the ACID to be used. CA Top Secret supports two system variables (system id and application id) that can be used to select the ACID. Sites can also define their own variables to be used as selection criteria. Criteria data is stored in a CRITMAP record in the SDT. CERTMAP and CRITMAP records are created with the TSS ADD command.

Directory Concepts

When a certificate name filter is defined, the information is stored in a CERTMAP record in the SDT on the security file. The filter definition specifies the significant portion of the issuer's or subject's distinguished name used to associate an ACID with a certificate.

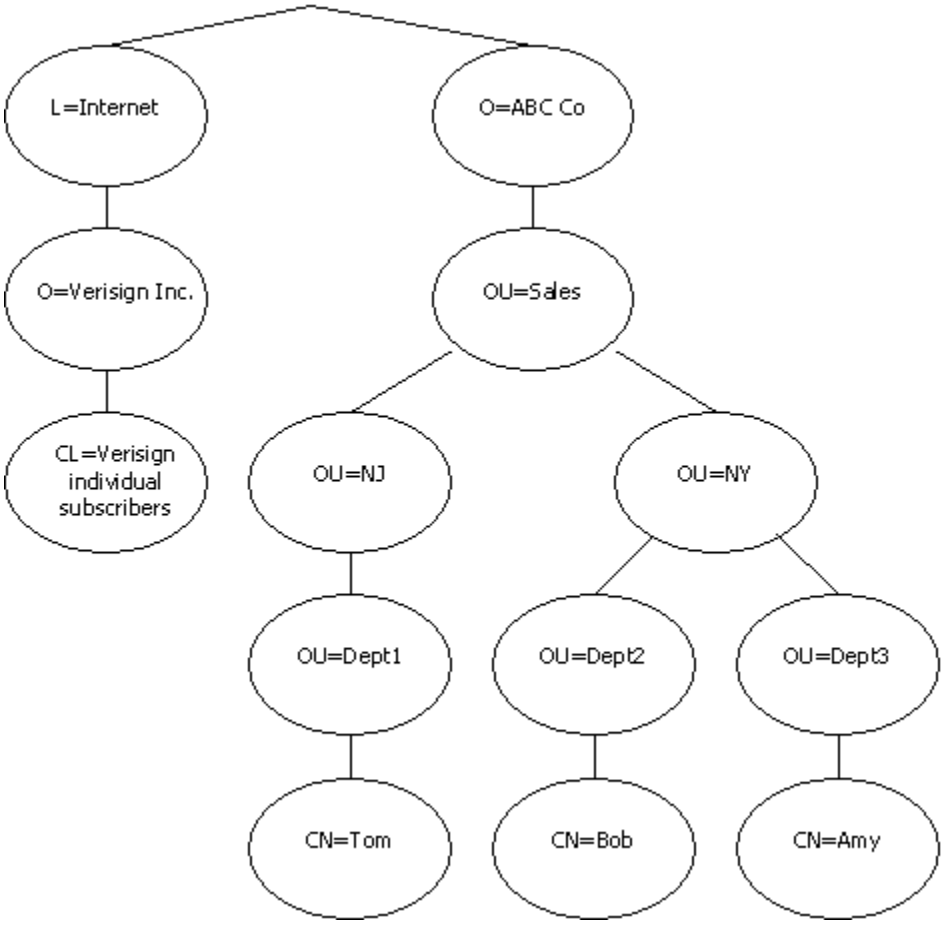
Additional criteria can be specified to identify the ACID used. The system ID and application ID system variables can be used to select the ACID. Sites can define their own variable for selection criteria.

Criteria data is stored in a CRITMAP record in the SDT. CERTMAP and CRITMAP records are created with the TSS ADD command.

To understand how certificate name filtering works, it is important to know directory concepts and the models described by the X.500 standard. The subject's and issuer's distinguished names on a certificate identify the subject's or issuer's location in an X.500 directory information tree.

The following diagram is an example of a directory tree:

Equation 1: Shows an Example Directory Tree


L=Internet
O=ABC Co
O=Verisign Inc.
OU=Sales
CL=Verisign individual subscribers
OU=NJ
OU=NY
OU=Dept1
OU=Dept2
OU=Dept3
CN=Tom
CN=Bob
CN=Amy

In this X.500 directory information tree, Amy's path name would be:

/O=ABC Co/OU=Sales/OU=NY/OU=Dept3/CN=Amy

Or, written in the address form used by CA Top Secret:

O=ABC Co.OU=Sales.OU=NY.OU=Dept3.CN=Amy

The nodes in this tree structure show that Amy works in department Dept3 in New York in the Sales division of the ABC Co company. A user's location in the hierarchy determines the access to resources that they have.

CA Top Secret supports this tree structure. ACIDs can be assigned to each level at which you want to group users or they can be assigned at just one level.

For example, node OU=Dept2 could be assigned to acid NYDEPT2 and OU=Dept3 could be assigned to NYDEPT3. When a user enters the system by presenting a certificate, CA Top Secret determines which ACID to assign by matching the subject's distinguished name to a node name.

If Amy entered the system with a certificate with a subject distinguished name, she would be assigned ACID NYDEPT3:
O=ABC Co.OU=Sales.OU=NY.OU=Dept3.CN=Amy

If ACID AMYUSR was assigned to node CN=Amy, she would be assigned ACID AMYUSR since that is a more specific match. Mapping is also done using the issuer name since two different certificate authorities can issue a certificate with the same subject name. ACID assignment can be based on a combination of subject name and issuer name, only a subject name or only an issuer name. Both full path names and partial path names can be defined.

You can use additional criteria (such as application ID or system ID) to select the ACID to be assigned to a certificate.

Certificate Name Filter Management

Use the ADD, REMOVE, REPLACE, and LIST commands to manage certificate name filters. The ACID specified on the command identifies the user to be assigned if the filter is matched. The MULTIID ACID indicates that additional criteria select the ACID.

To add a name filter, enter the command:

```
TSS ADD(userid) CERTMAP(recid)
                SDNFILTR(subject-dist-name-filter)
                IDNFILTR(issuer-dist-name-filter)
                CRITERIA(criteria-name-template)
                LABLCMAP(32 byte label)
                DCDSN(data set name)
                PKCSPASS('PKCSPASS PASSWORD')
                TRUST|NOTRUST
```

CERTMAP

Specifies a unique record identifier.

SDNFILTR

Specifies the significant portion of the subject's distinguished name that is to be used as a filter when associating an ACID with a certificate. The value specified for SDNFILTR must begin with a prefix found in the following list, followed by an equal sign (X'7E'). Each component should be separated by a period (X'4B'). The case, blanks, and punctuation displayed when the digital certificate information is listed must be maintained in the SDNFILTR. Since digital certificates only contain characters available in the ASCII character set, the same characters should be used for the SDNFILTR value.

For example: SDNFILTR('OU=BobsAcc')

Valid prefixes for SDNFILTR and IDNFILTR are:

- COUNTRY C=
- STATE/PROVINCE ST=
- LOCALITY L=
- ORGANIZATION O=
- ORGANIZATIONAL UNIT OU=
- TITLE T=
- COMMON NAME CN=
- DOMAIN COMPONENT DC=
- POSTAL CODE PC=
- EMAIL E=
- STREET NAME STREET=

- USERID UID=

IDNFILTR

Specifies the significant portion of the issuer's distinguished name that is to be used as a filter when associating an ACID with a certificate. The value specified for IDNFILTR should begin with a prefix found in the list above and must be followed by an equal sign (X'7E'). Each component should be separated by a period (X'4B'). The case, blanks, and punctuation displayed when the digital certificate information is listed must be maintained in the IDNFILTR. Since digital certificates only contain characters available in the ASCII character set, the same characters should be used for the IDNFILTR value.

For example: IDNFILTR('OU=Class 1 Certificate.0=BobsCertAuth')

CRITERIA

Is specified with the MULTIID ACID to identify variable data in addition to SDNFILTR and IDNFILTR. Criteria defined by CA Top Secret are CNFAPP and SYSID. Users can also define their own variables.

LABLCMAP

Specifies the label to be associated with the certificate name filter. It can contain embedded blanks and mixed-case characters, and is stripped of leading and trailing blanks. If a single quotation is intended to be part of the label-name, you must use two single quotation marks together for each single quotation mark within the string, and the entire string must then be enclosed within single quotation marks.
Range: Up to 32 characters.

DCDSN

Specifies the name of a data set that contains a digital certificate. The SDNFILTR or IDNFILTR data must match a portion of the subject/issuer's distinguished name extracted from the certificate. The distinguished name from the point of the match to the end of the name is used as the filter data.

PKCSPASS

The PKCS-password is case sensitive and can contain blanks.

Range: Up to 255 characters

Important! The password associated with PKCS#12 certificates are not viewable. It is the CA Top Secret administrator's responsibility to keep track of the PKCS#12 password that is assigned to the digital certificate.

TRUST|NOTRUST

When specified it indicates whether this mapping can be used to associate a userid to a certificate presented by a user accessing the system. If neither TRUST nor NOTRUST is specified, the default is NOTRUST.

Criteria Map Management

When the ACID is MULTIID and the CRITERIA keyword is specified on the TSS ADD CERTMAP command, criteria data must be defined in CRITMAP records to identify the ACID to be associated with a certificate. The ACID name on the CRITMAP record identifies the user when the filter that matched the certificate was for ACID MULTIID.

Use the ADD, REMOVE, REPLACE, and LIST commands to manage criteria maps.

This command has the following format:

```
TSS ADD(userid) CRITMAP(recid)
                {SYSID(system identifier)}
                {CNFAPP(application name)}
                {CNFUVAR(site variable list)}
```

Userid

Name of the ACID to be associated with this filter.

CRITMAP

Unique 8-byte record identifier.

SYSID

The system identifier. Can contain an asterisk (*) for masking.
Range: Up to 4 characters.

CNFAPP

The application variable. Can contain an asterisk (*) for masking.

Range: Up to 8 characters.

CNFUVAR

A list of application-defined variables that are defined as CRITERIA keyword data.
Range: Up to 255 uppercase characters.

Certificate Name Filter Scenarios

The following CA Top Secret command examples are based on the tree directory structure.

Examples: certificate name filters

In this example, users enter the system with a certificate subject that starts with:

```
OU=NJ.OU=Sales.O=ABC Co
```

These users are assigned ACID NJDEPT1 if the certificate was issued by the VeriSign certificate authority. If the subject matched but the certificate was issued by another certificate authority the user is assigned ACID NJDFLT.

```
TSS ADD(NJDEPT1) CERTMAP(NJMAP1)
                 LABLCMAP('NJ Dept 1 Map')
                 TRUST
                 IDNFILTR('OU=VeriSign Class 1 Individual
                          Subscriber.O=VeriSign, Inc.L=Internet')
                 SDNFILTR('OU=NJ.OU=Sales.O=ABC Co')

TSS ADD(NJDFLT) CERTMAP(NJDFLT)
                LABLCMAP('NJ Default user')
                TRUST
                SDNFILTR('OU=NJ.OU=Sales.O=ABC Co')
```

In this example, users enter the system with a certificate subject that starts with:

```
OU=Dept3.OU=NY.OU=Sales.O=ABC Co
```

These users are assigned ACID NYDEPT3.

```
TSS ADD(NYDEPT3) CERTMAP(NYMAP3)
                 LABLCMAP('NY Dept 3 Map')
                 TRUST
                 SDNFILTR('OU=Dept3.OU=NY.OU=Sales.O=ABC Co')
```

In this example additional criteria (in this case application id) decide which ACID to assign. Users in NY sales department Dept2 that handle corporate accounts (they use application BUSINESS to access the system) is assigned ACID NYDEPT2B and users that handle retail accounts (they use application RETAIL to access the system) is assigned ACID NYDEPT2R.

The special ACID name of MULTIID along with the CRITERIA parameter tells CA Top Secret that if the subject and/or the issuer name information matches, then search the CRITMAP records for a match on application name before assigning an ACID to the user.

```
TSS ADD(MULTIID) CERTMAP(NYMAP2)
                 LABLCMAP('NY Dept 2 Map')
```

```
                    TRUST
                    SDNFILTR('OU=Dept2.OU=NY.OU=Sales.O=ABC Co')
                    CRITERIA(CNFAPP=&CNFAPP)

TSS ADD(NYDEPT2B) CRITMAP(NYCRIT2B)
                   CNFAPP(BUSINESS)

TSS ADD(NYDEPT2R) CRITMAP(NYCRIT2R)
                   CNFAPP(RETAIL)
```

List Filtering Information

To list all the ACIDs and their CERTMAPS associated with them enter:

```
TSS LIST(SDT) CERTMAP(ALL)
```

To list all the ACIDs and their CRITMAP enter:

```
TSS LIST(SDT) CRITMAP(ALL)
```

To list the associated CERTMAP for a specific ACID enter:

```
TSS LIST(USER01) CERTMAP(MAP0001)
```

The command must contain the name of the CERTMAP already associated with the ACID.

To list the associated SEGMENT information for a specific ACID enter:

```
TSS LIST(USER01) SEGMENT(CMAPDATA)

TSS LIST(USER01) SEGMENT(CRITDATA)

TSS LIST(USER01) SEGMENT(ALL)
```

To list the associated DIGTCERT for a specific ACID enter.

```
TSS LIST(USER01) DIGICERT(CERT001)

TSS LIST(USER01) KEYRING(ACCTRING)
```

The command must contain the name of the DIGTCERT already associated with the ACID.

Init ACEE Changes for Search Sequence

InitACEE searches through the CERTMAP records. If an individual DIGICERT record does not match the incoming certificate:

- Check for a match on full issuer name and a full subject name.
- If the test does not find a match, take off the most specific piece of the SDN and test the entry again.
- Continue to take off pieces of the SDN until a match or until the end of the entry.
- If no match is found try again using the next entry with both IDNFILTR and SDNFILTR.
- If no more records exist with both fields or there are no more matches on the IDNFILTR, search through the records with only SDNFILTR. Start with the full SDN and continue to take pieces off until a match or until the end of the SDN.
- If there is no match on any of the records containing only SDNFILTR, look through the entries containing only IDNFILTRs.
- If no record matches the search criteria, an error code is returned.
- If a match is found, determine what ACID name should be returned. If MULTIID and CRITERIA were not specified, then the ACID name in the table is returned. If CRITERIA is available, find the CRITMAP record that matches the CRITERIA specified.

Search Sequence Scenario

In this example, the following records exist and all are trusted. They are listed in the order in which they are grouped in the search table.

```
CERTMAP(MAP001) ACID(NJDEPT1)
IDNFILTR(OU=Verisign Class 1 Individual Subscriber.O=Verisign,Inc.L=Internet)
SDNFILTR(OU=DEPT1.OU=NJ.OU=Sales.O=ABC Co)

CERTMAP(MAP002) ACID(NJDEPTX)
IDNFILTR(O=Verisign,Inc.L=Internet)
SDNFILTR(OU=Sales.O=ABC Co)

CERTMAP(MAP003) ACID(NYDEPT2)
SDNFILTR(OU=DEPT2.OU=NY,OU=Sales.O=ABC Co)

CERTMAP(MAP004) ACID(NYDEPT3)
SDNFILTR(OU=DEPT3.OU=NY,OU=Sales.O=ABC Co)

CERTMAP(MAP005) ACID(ABCDEPT)
SDNFILTR(OU=Sales.O=ABC Co)

CERTMAP(MAP006) ACID(ABCTECH)
SDNFILTR(OU=R&D.O=ABC Co)

CERTMAP(MAP007) ACID(MULTIID)
IDNFILTR(O=Verisign,Inc.L=Internet)
CRITERIA(CNFAPP=&CNFAPP)

CRITMAP(CRT001) ACID(ABCCUST)
CNFAPP(ABCINET)

CRITMAP(CRT002) ACID(ABCDFLT)
CNFAPP(*)
```

A certificate is presented by a user whose distinguished name is: CN=Bill,OU=Dept4,OU=PA,OU=Sales,O=ABC Co. The issuer's distinguished name contains information that is not VeriSign. The process to search for this certificate is:

- The first two entries do not match-the section ends without an IDNF.
- Loop through the SDNFs checking for a match.
- Take off the CN from the certificate distinguished name and compare the rest of the certificate distinguished name against the SDNF.
- The sections starting with OU=Dept4 and OU=PA do not match.
- The section starting with OU=Sales matches and the ABCDEPT ACID is assigned.

A user presents a certificate issued by VeriSign but not for ABC Co. There is a match on CERTMAP MAP007, based on the IDNF information. Then search the CRITMAP records for a matching CNFAPP. If the CNFAPP was ABCINET, then ACID ABCCUST is assigned. All other applications are assigned the default ACID ABCDFLT.

Using MQ WebSphere with CA Top Secret Certificates

You can use certificates generated and/or stored by CA Top Secret to secure MQ WebSphere. The certificates can be:

- Signed
- Self-signed by CA Top Secret

Example: MQ WebSphere with CA Top Secret Self-Signed Certificates

In this example MQ WebSphere is set up with CA Top Secret generated self-signed certificates:

1. Enter the command:

```
TSS GENCERT(MQCHIN1) DIGICERT(MCI1CERT)
                     SUBJECTN('o="COMPANYA"
                              CN=" MQCHIN1 selfsigned cert"
                              OU="SYSTEMSDEPT" C="US" ')
                     LABLCERT('ibmWebSphereMQCSQ1')
```

   A self signed certificate is generated where:

   - MQCHIN1 is the MQ Channel Initiator region ACID.
   - MCI1CERT is the digital certificate name.
   - LABLCERT 'ibmWebSphereMQ*xxxx*' where '*xxxx*' is the MQ channel initiator, and the certificate label.

2. Enter the commands:

```
TSS ADD(dept) IBMFAC(IRR.DIGTCERT) <---(skip if previously done)

TSS PERMIT(acid) IBMFAC(IRR.DIGTCERT.LISTRING)
                 ACCESS(UPDATE)

TSS PERMIT(acid) IBMFAC(IRR.DIGTCERT.LIST)
                 ACCESS(UPDATE)

TSS PERMIT(acid) IBMFAC(IRR.DIGTCERT.GENCERT)
                 ACCESS(UPDATE)
```

   The acid is authorized to read digital certificates.

   Note: If the owner of the client/personal certificate is ACID CERTSITE, specify ACCESS(CONTROL) on the PERMIT commands.

3. Enter the command:

```
TSS ADD(MQCHIN1) KEYRING(MCI1RING)
                 LABLRING(MCI1RING)
```

   The MQ Channel Initiator's KEYRING is created where MCI1RING is the KEYRING name and the KEYRING label name.

4. Enter the command:

```
TSS ADD(MQCHIN1) KEYRING(MCI1RING)
                 RINGDATA(MQCHIN1,MCI1CERT)
                 USAGE(PERSONAL)
```

   The certificate is added to the KEYRING.

5. Enter the MQ WebSphere command:

```
ALTER QMGR SLKEYR(MCI1RING)
```

   The queue manager's KEYRING is specified.

6. Enter the command:

```
TSS EXPORT(MQCHIN1) DIGICERT(MCI1CERT)
                    DCDSN('MQCHIN1.CERT')
```

   The certificate is exported to a dataset called 'MQCHIN1.CERT'.

7. Send the certificate to the client.

8. Recycle the MQ Websphere address space.

Example: MQ WebSphere with CA Top Secret Generated Signed Certificates

This example sets up MQ WebSphere with CA Top Secret generated signed certificates:

1. Enter the command:

```
TSS GENCERT(MQCHIN1) DIGICERT(MCI1CERU)
                     SUBJECTN('O="COMPANYA" CN=" MQCHIN1
                              cert" OU="SYSTEMSDEPT" C="US" ')
                     LABLCERT('MCI1CERU')
```

   A certificate is generated where:

   - MQCHIN1 is the MQ Channel Initiator region ACID
   - MCI1CERU is the digital certificate and LABLCERT

2. Enter the commands:

```
TSS ADD(dept) IBMFAC(IRR.DIGTCERT) <---(skip if previously done)
TSS PER(acid) IBMFAC(IRR.DIGTCERT.LISTRING)
              ACCESS(UPDATE)
TSS PER(acid) IBMFAC(IRR.DIGTCERT.LIST)
              ACCESS(UPDATE)
TSS PER(acid) IBMFAC(IRR.DIGTCERT.GENCERT)
              ACCESS(UPDATE)
```

   The ACID is authorized to read digital certificates.

3. Enter the command:

```
TSS GENREQ(MQCHIN1) DIGICERT(MCI1CERU)
                    DCDSN('MQCHIN1.UNSIGNED.CERT')
```

   The certificate is copied to a dataset in PKCS#10 format.

4. Send the certificate to the Certificate Authority to be signed.
5. Send the signed certificate to a dataset.
6. Enter the command:

```
TSS ADD(MQCHIN1) DIGICERT(MCI1CERS)
                 DCDSN('MQCHIN1.SIGNED.CERT')
                 LABLCERT('ibmWebSphereMQCSQ1')
                 TRUST
```

   The certificate is stored on the CA Top Secret Security File where:

   - 'MQCHIN1.SIGNED.CERT' is the dataset
   - MCI1CERS is the new DIGICERT certificate name

   Note: LABLCERT must be 'ibmWebSphereMQ*xxxx*' where '*xxxx*' is the MQ channel initiator.

7. Enter the command:

```
TSS ADD(MQCHIN1) KEYRING(MCI1RING)
                 LABLRING(MCI1RING)
```

   The MQ Channel Initiator's KEYRING is created.

8. Enter the command:

```
TSS ADD(MQCHIN1) KEYRING(MCI1RING)
                 RINGDATA(MQCHIN1,MCI1CERS)
                 USAGE(PERSONAL)
                 DEFAULT
```

   The certificate is added to the KEYRING.

9. If you are using the Certificate Authorities' public key, skip to step 12.

10. Enter the command:

```
TSS EXPORT(MQCHIN1) DIGICERT(MCI1CERS)
                    DCDSN('MQCHIN1.SIGNED.CERT')
                    LABLCERT(MCI1CERS)
```

   The certificate is exported to the 'MQCHIN1.SIGNED.CERT'.dataset.

11. Send the certificate to the client.

12. Enter the MQ WebSphere command:

```
ALTER QMGR SLKEYR(MCI1RING)
```

   The queue manager's KEYRING is specified.

13. Send the Certificate Authority to the mainframe.

14. Enter the command:

```
TSS ADD(CERTAUTH) DIGICERT(MCI1CA)
                  DCDSN('MQCHIN1.CERT.AUTH')
```

   The Certificate Authority is added to CERTAUTH.

15. Enter the command:

```
TSS ADD(MQCHIN1) KEYRING(MCI1RING)
                 RINGDATA(CERTAUTH,MCI1CA)
                 USAGE(CERTAUTH)
```

   The Certificate Authority is added to the MQ Channel Initiator's KEYRING.

16. Recycle the MQ Websphere address space.

FTP Server and Client Authentication

OE/FTP is an OMVS application that executes under USS to facilitate file transfers throughout a TCP/IP network. OE/FTP is packaged with TCP/IP OE Application Services.

CA Top Secret Digital Certificates are a secure way to identify servers and clients when using OE/FTP.

FTP Server Authentication-Mainframe to PC

Use CA Top Secret Digital Certificates as a secure way to identify users when using OE/FTP services.

To authenticate a Mainframe FTP Server from a FTP client on the PC

1. Enter the command:

   ```
   TSS GENCERT(FTPS)  DIGICERT(FTPSCERT)
   ```

   The FTP server's certificate is generated and added to the FTP region ACID FTPS.

2. Enter the command:

   ```
   TSS ADD(FTPS) KEYRING(FTPSRING)
                 LABLRING(FTPSRING)
   ```

   The FTP server's KEYRING is created.

   Note: There are no blank spaces in the LABLRING.

3. Enter the command:

   ```
   TSS ADD(FTPS) KEYRING(FTPSRING)
                 RINGDATA(FTPS,FTPSCERT)
                 DEFAULT
                 USAGE(PERSONAL)
   ```

   The FTP server's certificate is added to the FTP server's KEYRING.

4. Enter the command:

   ```
   TSS EXPORT(FTPS) DIGICERT(FTPSCERT)
       DCDSN('FTPS.SERVER.CERT')
   ```

   The FTP server's certificate is copied to a dataset. The dataset is automatically created and cataloged.

5. Use your FTP product to copy the FTP server's certificate FTPS.SERVER.CERT to the FTP client's Trusted Authorities database.

6. Enter the commands:

```
TSS PER(FTPS) IBMFAC(IRR.DIGTCERT.GENCERT) ACC(UPDATE|CONTROL)
TSS PER(FTPS) IBMFAC(IRR.DIGTCERT.LISTRING) ACC(UPDATE|CONTROL)
TSS PER(FTPS) IBMFAC(IRR.DIGTCERT.LIST) ACC(UPDATE|CONTROL)
TSS PER(USRA) IBMFAC(IRR.DIGTCERT.GENCERT) ACC(UPDATE|CONTROL)
TSS PER(USRA) IBMFAC(IRR.DIGTCERT.LISTRING) ACC(UPDATE|CONTROL)
TSS PER(USRA) IBMFAC(IRR.DIGTCERT.LIST) ACC(UPDATE|CONTROL)
```

   Use ACC(CONTROL) *only* if CERTSITE is the owner of the certificate.

   The FTP ACID is permitted to the SSL KEYRING, certificates, and mappings.

7. Open IBM's FTPS.DATA member for editing and add the following IBM FTP parameters:

   - KEYRING FTPSRING
   - SECURE_LOGIN NO_CLIENT_AUTH
   - SECURE_FTP REQUIRED
   - AUTH TLS

   The keyring name is established with FTP, client authentication is disabled, and FTP server authentication is activated.

For more information on activating digital certificates with FTP, see the IBM documentation.

FTP Client Authentication-Mainframe to PC (Optional)

FTP client authentication is not required for FTP server authentication. If you choose to use FTP client authentication, FTP server authentication must be working.

To authenticate a PC FTP client from the mainframe FTP Server

1. Export the FTP server's certificate FTPS.SERVER.CERT to the PC and bring it into the FTP client's Trusted Authorities database.
2. Enter the command:

   ```
   TSS GENCERT(USERA) DIGICERT(USRACERT)
                      SUBJECTN('o=”COMPANYA” CN=”USERA selfsigned ftp cert” -
                               OU=”SYSTEMSDEPT” C=”US”')
                      LABELCERT('USERA CERT')
   ```

   The FTP client certificate is generated.
3. Enter the command:

   ```
   TSS ADD(USERA) KEYRING(USRARING)
                  LABLRING(USRARING)
   ```

   The KEYRING for the FTP client ACID is created.
4. Enter the command:

   ```
   TSS ADD(USERA) KEYRING(USRARING)
                  RINGDATA(USERA,USRACERT)
                  DEFAULT USAGE(PERSONAL)
   ```

   The FTP client's certificate is added to the FTP *client's* KEYRING.
5. Enter the command:

   ```
   TSS ADD(FTPS) KEYRING(FTPSRING)
                 RINGDATA(USERA,USRACERT)
                 USAGE(PERSONAL)
   ```

   The FTP client's certificate is added to the FTP *server's* KEYRING.
6. Enter the command:

   ```
   TSS EXPORT(USERA) DIGICERT(USRACERT)
                     DCDSN(USERA.CERT)
   ```

   The FTP client's certificate is export to the USERA.CERT dataset. This is automatically created and cataloged by CA Top Secret.
7. Use your FTP product to export the USERA.CERT client certificate to the PC and bring it into the FTP client's Trusted Authorities database.
8. Open IBM's FTPS.DATA member and add the following FTP parameter:

   ```
   SECURE_LOGIN VERIFY_USER
   ```

   FTP client authentication is activated.

FTP Server Authentication-Mainframe to Mainframe

CA Top Secret Digital Certificates are a secure way to identify users when using OE/FTP services.

To authenticate a mainframe FTP Server from a FTP client on the mainframe

1. Enter the command:

   ```
   TSS GENCERT(FTPS) DIGICERT(FTPSCERT)
   ```

   The FTP server's certificate is generated and added to the FTP region ACID FTPS.

2. Enter the command:

   ```
   TSS ADD(FTPS) KEYRING(FTPSRING)
                 LABLRING(FTPSRING)
   ```

   Create the FTP server's KEYRING.

   Note: There are no blank spaces in the LABLRING.

3. Enter the command:

   ```
   TSS ADD(FTPS) KEYRING(FTPSRING)
                 RINGDATA(FTPS, FTPSCERT)
                 DEFAULT
                 USAGE(PERSONAL)
   ```

   The FTP server's certificate is added to the FTP server's KEYRING.

4. Enter the command:

   ```
   TSS EXPORT(FTPS) DIGICERT(FTPSCERT)
                    DCDSN('FTPS.SERVER.CERT')
   ```

   The FTP server's certificate is copied to a dataset. This dataset does not have to be formatted and is automatically created and cataloged by CA Top Secret.

5. Use your FTP product to export the FTPS.SERVER.CERT server certificate to the other mainframe and add it to the FTP client's KEYRING.

   a. If using CA Top Secret, enter the command:

      ```
      TSS ADD(USERA) KEYRING(USRARING)
                     RINGDATA(FTPS,FTPSCERT)
                     DEFAULT
                     USAGE(PERSONAL)
      ```

      The FTP server's certificate is copied from the FTP client's KEYRING.

   b. If using another product, refer to their manual for information on how to add the certificate to the user's KEYRING.

6. Enter the commands:

```
TSS PER(FTPS) IBMFAC(IRR.DIGTCERT.GENCERT) ACC(UPDATE|CONTROL)
TSS PER(FTPS) IBMFAC(IRR.DIGTCERT.LISTRING) ACC(UPDATE|CONTROL)
TSS PER(FTPS) IBMFAC(IRR.DIGTCERT.LIST) ACC(UPDATE|CONTROL)
TSS PER(USRA) IBMFAC(IRR.DIGTCERT.GENCERT) ACC(UPDATE|CONTROL)
TSS PER(USRA) IBMFAC(IRR.DIGTCERT.LISTRING) ACC(UPDATE|CONTROL)
TSS PER(USRA) IBMFAC(IRR.DIGTCERT.LIST) ACC(UPDATE|CONTROL)
```

   Use ACC(CONTROL) *only* if CERTSITE is the owner of the certificate.

   The FTP server's region ACID and the FTP client ACID are permitted to the SSL KEYRING, certificates, and mappings.

7. Open the IBM's FTPS.DATA member and add the following parameters :

   - KEYRING FTPSRING
   - SECURE_LOGIN NO_CLIENT_AUTH
   - SECURE_FTP REQUIRED
   - AUTH TLS

   The keyring name is established with FTP, client authentication is disabled, and FTP server authentication is activated.

FTP Client Authentication-Mainframe to Mainframe (Optional)

FTP client authentication is not required for FTP server authentication. If you choose to use FTP client authentication, FTP server authentication must be working.

To authenticate a mainframe FTP client from a mainframe FTP Server

1. Verify that FTP server authentication is working.
2. Enter the command:

```
TSS GENCERT(USERA) DIGICERT(USRACERT)
                   SUBJECTN('o=”COMPANYA” CN=”USERA
                            selfsigned ftp cert” OU=”SYSTEMSDEPT” C=”US”')
                   LABELCERT('USERA CERT')
```

   The FTP client certificate is generated.
3. Enter the command:

```
TSS ADD(USERA) KEYRING(USRARING)
               LABLRING(USRARING)
```

   The KEYRING for the FTP client ACID is created.
4. Enter the command:

```
TSS ADD(USERA) KEYRING(USRARING)
               RINGDATA(USERA,USRACERT)
               DEFAULT
               USAGE(PERSONAL)
```

   The FTP client's certificate is added to the FTP client's KEYRING.
5. Enter the command:

```
TSS ADD(FTPS) KEYRING(FTPSRING)
              RINGDATA(USERA,USRACERT)
              DEFAULT
              USAGE(PERSONAL)
```

   The FTP client's certificate is added to FTP the server's KEYRING with CERTAUTH authority.
6. Enter the command:

```
TSS ADD(USERA) KEYRING(USRARING)
               RINGDATA(FTPS,FTPSCERT)
               USAGE(PERSONAL)
```

   The FTP server's certificate is copied to the FTP client's KEYRING.
7. Open IBM's FTPS.DATA member and add the following parameter:

```
FTP SECURE_LOGIN VERIFY_USER
```

   FTP client authentication is activated.

Chapter 6: Kerberos

This section contains the following topics:

About Kerberos

Kerberos, uses CA Top Secret to store and administer information about principals and realms.

The KERB segment of the user record stores information about Network Authentication and Privacy Service principals on your local system. The SDT record KERBLINK lets you map principals to CA Top Secret user IDs on your system. The SDT record REALM defines the local Network Authentication and Privacy Service realm and its trust relationships with foreign realms.

If you need to define many local principals, you may need to define a VSAM file to hold the records. For information, see the *Installation Guide*.

Authentication of Principals

Kerberos for z/OS verifies requests as a trusted third party authentication service. Using conventional shared key cryptography, Kerberos confirms the identities of principals (users):

- Without relying on authentication by the host operating system
- Without basing trust on host addresses
- Without necessitating physical security of all hosts on the network
- Under the premise that packets traveling along the network can be read, changed, and inserted at will

Electronic Tickets

Kerberos uses electronic tickets to authenticate a user to a server. The ticket is an encrypted message containing the names of the user and server, the user's network address, a time stamp, and a session key. A ticket is for a single server, a single user, and a certain period of time.

The user can use the ticket to access the server as many times as desired until the ticket expires. The user cannot decrypt the ticket. Nobody listening in on the network can read or modify the ticket as it passes through the network without detection or invalidation.

Kerberos Verification Process

The Kerberos protocol involves two servers, the Kerberos Authentication Server and one or more Ticket-Granting Servers (TGSs).

The process is:

- The user requests a ticket from the Kerberos Authentication Server to the Kerberos TGS. This request takes the form of a message containing the user's name and the name of their TGS.
- The Authentication Server looks up the user in its database and generates an encrypted session key to be used between the user and the TGS.
- The Authentication Server creates a ticket-granting ticket (TGT) for the user to present to the TGS and encrypts the TGT using the TGS's secret key.
- The Authentication Server sends both encrypted messages to the user.
- The user decrypts the first message and recovers the session key.
- The user creates an authenticator consisting of his or her name and address and a time stamp encrypted with the session key.
- The user sends a request to the TGS for a ticket to a particular target server. This request contains the name of the server, the TGT received from Kerberos, and the encrypted authenticator.
- The TGS decrypts the TGT and then uses the session key included in the TGT to decrypt the authenticator.
- The TGS compares information. If everything matches, it allows the request to proceed.
- The TGS creates a new session key for the user and target server and incorporates this key into a valid ticket for the user to present to the server. The TGS also encrypts the new user-target session key using the session key shared by the user and the TGS.
- The TGS sends both messages to the user.
- The user decrypts the message and extracts the session key.
- The user creates a new authenticator encrypted with the user-target session key that the TGS generated.
- To request access to the target server, the user sends the ticket received from Kerberos and the encrypted authenticator

  Because this authenticator contains plain text encrypted with the session key, it proves that the sender knows the key. Encrypting the time of day prevents an eavesdropper who records both the ticket and the authenticator from replaying them later.
- The target server decrypts and checks the ticket and the authenticator, also checking the user's address and the time stamp.

- For applications that require mutual authentication, the server sends the user a message consisting of the time stamp plus 1, encrypted with the session key to prove to the user that the server knew its secret key and was able to decrypt the ticket and the authenticator.

Realms

The Kerberos protocol functions across organizational boundaries. An organization running on a Kerberos server must establish its own realm. A client's name contains the name of the realm in which a client is registered, so that the realm name can be used by the application server to determine whether to grant a request.

Inter Realm Keys

With the establishment of inter-realm keys, the administrators of the two realms can permit a client authenticated in one realm to use its credentials in the other realm. Exchanging inter-realm keys registers the ticket-granting service of each realm as a principal in the other. A client can then procure a ticket-granting ticket for the remote realm's ticket-granting service from its local ticket-granting service. Tickets distributed to a service in the remote realm indicate that the client was authenticated from another realm.

This procedure can be used to authenticate throughout an organization across multiple realms. To construct an authentication path to a foreign realm, the local realm must share an inter-realm key with the target realm or with an intermediate realm that communicates with the target realm or with another intermediate realm.

Hierarchical Realms

Realms are often organized hierarchically. A realm shares a key with its parent and different key with each child. In a hierarchical organization, an authentication path can be easily established if an inter-realm key is not shared by two realms. If a hierarchical organization is not in place, referring to a database in order to build an authentication path between realms may be required.

Although realms are often hierarchical, intermediate realms may be overridden, resulting in cross-realm authentication through alternate authentication paths. The end-service must know which realms were transited when determining how much confidence to have in the authentication process. To aid this process, a field in each ticket includes the realm names that helped authenticate the client.

Using Kerberos

New records in the SDT are defined to provide REALM definitions to describe the local and foreign environments, which the local server is expected to recognize. Local ACIDs are equipped with an additional KERB segment, containing Kerberos information, and are mapped for fast access in the SDT. Foreign ACIDs are linked to local ACIDs through additional SDT definitions.

In this guide, the resource HFSSEC is used for UNIX file security. The IBMFAC(SUPERUSER.) resource can be substituted in the following examples.

Local Server Configuration

For information on installing the Kerberos Server, see the *z/OS Network Authentication Service Administration Guide*.

To implement the Kerberos Server SKRBKDC

1. Enter the command:

```
TSS CREATE(SKRBKDC) NAME('kerb server acid')
                    PASS(NOPW,0)
                    DEPT(sysdept)
                    FACILITY(BATCH,STC,OPENMVS)
                    SOURCE(INTRDR)
```

   A region ACID for the procedure is defined.

2. Enter the command:

```
TSS ADD(SKRBKDC) UID(0)
                 HOME(/etc/skrb/home/kdc)
                 OMVSPGM(/bin/sh)
                 GROUP(omvsgrp)
                 DFLTGRP(omvsgrp)
TSS PER(SKRBKDC) HFSSEC(/BIN.SH)
                 ACC(READ,EXEC)
TSS PER(SKRBDDC) HFSSEC(/ETC.SKRB)
                 ACC(READ)
```

   Permissions and keywords for the ACID are established.

3. Add any permissions needed as determined by the variables settings in the configuration file, /var/skrb/home/kdc/envar:

```
TSS PER(SKRBKDC) HFSSEC(nlspath)
                 ACC(READ,EXEC)
TSS PER(SKRBKDC) HFSSEC(nlslocale)
                 ACC(READ,EXEC)
```

   The installation defaults are:

```
nlspath: /USR.LPP.SKRB.LIB.NLS.MSG.EN_US$IBM$1047.SKR
nlslocale: /USR.LIB.NLS.LOCALE.EN_US
```

   This will differ if you apply a different language path (NLSPATH) in the configuration environment.

4. Determine the STDOUT and STDERR files specified in your file. In the Kerberos configuration file, the variables will contain the file names required.

```
EUV_SVC_STDOUT_FILENAME
EUV_SVC_STDERR_FILENAME
```

5. Enter the command:

```
TSS PER(SKRBKDC) HFSSEC(/VAR.SKRB.LOGS.)
                 ACC(ALL)
```

   Access is allowed to the STDERR and STDOUT files.

6. Enter the command:

```
TSS PER(SKRBKDC) HFSSEC(/VAR.SKRB.CREDS)
                 ACC(ALL)
```

   Read and write credentials are added to the server.

7. Enter the command:

```
TSS ADD(STC) PROCNAME(SKRBKDC)

             ACID(SKRBKDC)
```

The procedure is added to the STC.

Define Your Local Realm

To define the realms defined to your Kerberos configuration file to the security file, enter:

```
/etc/skrb/krb5.conf
```

The local realm is specified in your default_realm parameter. The local realm is always named KERBDFLT and must be defined before the local principals are defined.

Syntax

This command has the following format:

```
TSS ADDTO(SDT) REALM(KERBDFLT)
               REALMNAME(default_realm)
               MINTKTLF(Min-ticket-life)
               MAXTKTLF(Max-ticket-life)
               DEFTKTLF(Default-ticket-life)
               KERBPASS(Kerberos-password)
               CHKADDRS
```

REALM

KERBDFLT (required)

REALMNAME

Must be identical to configuration file default_realm.

CA Top Secret has simplified the REALMNAME specification. This is internally expanded to the following when used for run-time security checks:

`/…/default_realm/krbtgt/default_realm`

The REALMNAME is generally specified in the form of a first order web-address. Cannot include spaces (x'40') or the at-sign (x'7F').
Range: Up to 117 characters.

Note: Because the relationship between the REALMNAME and generating Kerberos tickets for principal users is based, in part, on the local REALMNAME, care must be taken when choosing a REALMNAME. Renaming the REALM should be avoided at all costs during Kerberos operations, trust relations in flight cause unpredictable effects.

MINTKTLF

Specifies the maximum ticket life in seconds, and is represented by a numeric value. Note that 0 is not a valid value. This keyword is only applicable when defining the KERBDFLT realm record. If MAXTKTLF is specified, DEFTKTLF and MINTKTLF must also be specified.

Range: 1 to 247483647

DEFTKTLF

Specifies the time intervals (in seconds) that a Kerberos generated ticket will remain active in the realm. If any of these intervals is specified, all must be specified. If no intervals are specified, tickets are not limited. MAXTKTLF >= DEFTKTLF >= MINTKTLF is enforced.
Range: 1 to 2**31-1 (2147483647)

KERBPASS

Specifies a password (alphanumeric) for the local realm. When the same realm name is used as a foreign realm in a foreign Kerberos system, the passwords must be identical. Passwords are case-sensitive and are maintained in the case in which they are entered. The KERBPASS bears no relationship to the password of the SKRBKDC region ACID.

Range: 1 to 8 characters

CHKADDRS

Enables address checking in tickets for the Kerberos server running on z/OS 1.13 and higher. This field can be enabled for the local realm only.

Default: NO CHKADDRS

Example: identify REALM name

In this example:

- The realm name KERBDFLT, identifies this record as the default realm record
- The Kerberos realm name is local.ca.com. This corresponds to the following fully qualified Kerberos security name which CA Top Secret formats automatically.

  `/…/local.ca.com/kerbtgt/local.ca.com`
- The Kerberos password of this realm is children
- The default ticket life is 10 hours, a minimum ticket life of 30 seconds, and a maximum ticket life of 24 hours.

```
TSS ADDTO(SDT) REALM(KERBDFLT)
               REALMNAME(local.ca.com)
               MINTKTLF(30)
                MAXTKTLF(86400)
               DEFTKTLF(36000)
               KERBPASS(children)
               CHKADDRS
```

Local Principals Definition

Local principals in Kerberos are user ACIDs capable of activity on and USS, who are able to initiate Kerberos commands in the system corresponding to the local realm. Kerberos local principals must possess at least the following with sufficient authority to allow access to the OMVS or ISHELL shells:

- UID
- GROUP
- DFLTGRP
- HOME
- OMVSPGM

When a local principal is defined, the affected ACID receives a KERB segment to hold relevant information, some of which will not be available for display. A corresponding KERBSEGM record will also be defined to the SDT. The label for the KERBSEGM record is the ACID to which the new segment has been added.

Define Kerberos Information

To define Kerberos information for the user enter:

```
TSS ADDTO(SDT) REALM(KERBDFLT)
               REALMNAME('Kerberos-realm-name')
               MINTKTLF(Min-ticket-life)
               MAXTKTLF(Max-ticket-life)
               DEFTKTLF(Default-ticket-life)
               ENCRYPT( '[DES | NODES] [,DESD | NODESD] [,DES3 | NODES3]
                       [AES128|NOAES128] [AES256|NOAES256]')
               KERBPASS(Kerberos-password)
               CHKADDRS
```

ACID

Specifies the security ACID to be defined with Kerberos information.

principal_name

Specifies the Kerberos Principal Name associated with this user. This information is added to the KERB segment of the user's security record. It is also added to the SDT KERBNAME record for high-speed cross-reference indexing. The KERBNAME specified must be unique for each user in the local realm. KERBNAME cannot be added to a PROFILE or GROUP ACID, nor can it be added to a hierarchy ACID. The fully qualified Kerberos principal name is formatted from the KERBDFLT REALMNAME and the KERBNAME principal_name.

The principal name cannot include spaces (x'40') or the "at" sign (x'7F').
Range: The combined length cannot exceed 240 characters.

```
/…/local_realm/principal_name
```

ENCRYPT

Specifies the level of encryption defined for your local realm. This must correspond to the encryption level selected in the krb5.config file described in IBM SecureWay® Security Server Network Authentication Service Administration. The options are:

DES | NODES

Indicates that single-DES encryption is set or not set for this realm.

DESD | NODESD

Indicates that double-DES encryption is set or not set for this realm.

DES3 | NODES3

Indicates that triple-DES encryption is set or not set for this realm.

AES128 | NOAES128

Indicates that AES 128 bit encryption is set or not set for this realm.

AES256 | NOAES256

Indicates that AES 256 bit encryption is set or not set for this realm.

interval

Specifies the maximum ticket life associated with tickets for this user. Sensible values for this parameter should not exceed MAXTKTLF for the REALM, and should not be exceeded by the REALM DEFTKTLF.
Range: 1 to 2**31 – 1.

CHKADDRS

Enables address checking in tickets for the Kerberos server running on z/OS 1.13 and higher. This field can be enabled for the local realm only.

Default: NO CHKADDRS

Examples: Kerberos

This example creates a KERBNAME field “john_doe” within a new KERB segment on “useracid”. Note that a MAXTKTLF operand is not supplied in this command.

```
TSS ADD(useracid) KERBNAME(john_doe)
```

This example displays the KERB segment:

```
TSS LIST(useracid) SEGMENT(KERB)
```

This example displays the corresponding KERBSEGM SDT record:

```
TSS LIST(SDT) KERBSEGM(useracid)
```

Local Environment Customization

Realms are defined in the SDT. The default_realm specification is also known as the “local” realm. The other realms defined in the configuration are known as “foreign” realms.

Users defined to Kerberos are not defined in the configuration file, but must be defined entirely through the Security File. Users defined in the local realm are known as “local” principals. Only local principals are allowed to initiate Kerberos commands from the local and Unix Systems Services. Users defined in a foreign realm are mapped in the SDT with a surrogate user in the local realm.

Password Change Server ACID

Kerberos requires a password change server ACID with a reserved local principal name. The only required ACID characteristic is that the user has the local principal name kadmin/changepw.

To define a USS capable user, enter:

```
TSS CREATE(KRBCHG) DEPT(sysdept)
                   NAME('KERBEROS PSWD/CHG')
                   PASS(pswd,0)
                   FAC(STC,BATCH)

TSS ADD(KRBCHG) UID(…)

TSS ADD(KRBCHG) GROUP(OMVSGRP)
                DFLTGRP(OMVSGRP)

TSS ADD(KRBCHG) HOME(/u/krbchg)
                OMVSPGM(/bin/sh)

TSS PER(KRBCHG) HFSSEC(/u.krbchg)
                ACC(READ,UPDATE,EXEC)

TSS PER(KRBCHG) HFSSEC(/BIN.SH)
                ACC(READ,EXEC)

TSS ADD(KRBCHNG) KERBNAME(kadmin/changepw)
```

Preparing Local Principal ACIDs

This section describes how to prepare your local principal ACIDs for Kerberos.

SYSEXEC Changes

Add the following data set to the TSO SYSEXEC DD statement of the Kerberos user's TSO procedure:

```
EUVF.SEUVFEXC
```

.profile Changes

Changes are required for Kerberos user's .profile file in their home directory. These changes may optionally be added to the /etc/.profile file. Place the following directory as the first directory in the PATH variable, so that it overrides DCE.

```
/usr/lpp/skrb/bin
```

It must be placed earlier in the path than the following to avoid DCE/Kerberos from overriding more recent Kerberos software.

```
/bin
Any /bin DCE subdirectory
```

Any Kerberos session variables which the user wishes to set differently than the values in the Kerberos environment file should be set in the user's .profile file. For information, consult IBM SecureWay Security Server Network Authentication Service Administration. If any file names are specified, the administrator must provide appropriate HFSSEC permissions in order to access them.

DCE Incompatibility

DCE commands conflict with Kerberos commands.To use these features change your environments as outlined in IBM SecureWay Security Server Network Authentication Service Administration.

Security Considerations

The process to assure the capability for the Kerberos kinit function is:

- Give the ACID standard OMVS attributes:

```
TSS ADD(local) UID(uid_number)
               GROUP(omvsgrp)
               DFLTGRP(omvsgrp)
               HOME(home_directory)
               OMVSPGM(program_directory)
```

- Give the ACID permission to change the mode of their own files:

```
TSS PERMIT(local) IBMFAC(BPX.CAHFS.FILE.MODE)
```

- Permit the ACID to create, alter, execute and read files in the home directory. Kerberos credentials will be created in this directory:

```
TSS PERMIT(local) HFSSEC(home_directory)
                  ACC(ALL)
```

- Permit the ACID must to READ and EXECUTE files in the program directory:

```
TSS PERMIT(local) HFSSEC(program_directory)
                  ACC(READ,EXEC)
```

- Permit access to Kerberos command files at run-time:

```
TSS PER(useracid) HFSSEC(/usr.lpp.skrb.bin)
```

- Permit access to Kerberos EXEC files:

```
TSS PER(useracid) DATASET(EUVF.SEUVFEXC)
                  ACC(READ)
```

Note: In these examples HFSSEC(ON) is set.

Map Foreign Environments

Corresponding to the local (default) realm and local principal users in the local system are the concepts of foreign realm and foreign principal user. The foreign realm is mapped into a web address, which the Kerberos server will contact for tickets. Foreign realms must be defined to the Kerberos configuration file. The corresponding SDT definition defines a trust relationship between the local web-address and the foreign web-address. The foreign principal definition defines a mapping from a user and a web-address into a local ACID. Foreign principals are not defined in the Kerberos configuration file.

Map Foreign Realms

To define foreign REALM commands to your Kerberos configuration file use:

```
/etc/skrb/krb5.conf
```

Foreign realm definitions are placed after the first realm section entry (which defines the Kerberos local realm). While the SDT REALM for the local realm is fixed as KERBDFLT, the administrator must select an 8-character label for the REALM which identifies it to CA Top Secret. Foreign REALMNAME is specified differently than local realms, as this operand represents a Kerberos trust relationship:

```
/…/local_realm/krbtgt/foreign_realm
/…/foreign_realm/krbtgt/local_realm
```

The ellipsis characters (…) are part of the Kerberos naming syntax. The two entries are necessary for the trust relationship and each entry has a different password. A pair of commands is required to define each direction of the Kerberos trust relationship at each local copy of CA Top Secret.

Example: REALM mapping

In this example, a trust relationship is established between TRUSTWORTHY.CA.COM with MAJESTERIAL.CLIENT.COM with the label MAJESTY in the SDT.

On the local system, define the following:

```
TSS ADD(SDT) REALM(majesty)
    REALMNAME('/…/trustworthy.ca.com/krbtgt/majesterial.client.com')
    PASSWORD(xylofone)

TSS ADD(SDT) REALM(trustyca)
    REALMNAME('/…/majesterial.client.com/krbtgt/trustworthy.ca.com')
    PASSWORD(marimba)
```

On the foreign system, a set of parallel definitions is required so that each connection in the conversation maintains identical passwords:

```
TSS ADD(SDT) REALM(kingart)
    REALMNAME('/…/trustworthy.ca.com/krbtgt/majesterial.client.com')
    PASSWORD(xylofone)

TSS ADD(SDT) REALM(troubador)
    REALMNAME('/…/majesterial.client.com/krbtgt/trustworthy.ca.com')
    PASSWORD(marimba)
```

The REALM operands are labels of convenience and do not have to match between the two systems. However, the password for each trust relationship must be identical for identical REALMNAME specifications.

Map Foreign Principal Names

Foreign principal names are defined on CA Top Secret to indicate the realm from which the foreign user will be mapped into a local ACID. The following format is used for this fully qualified reference:

```
/…/foreign_realm/foreign_principal_name
```

The foreign_principal_name should be defined in the foreign_realm as a local principal in that system. The foreign_principal_name need not be identical with its associated ACID in the foreign system.

The local ACID need not be defined as a Kerberos local principal. It serves as a surrogate for security activities by the foreign principal in the local environment.

Example: foreign principle mapping

In this example, the MAJESTERIAL.CLIENT.COM and COLLOSSAL.SUCCESS.COM foreign realms establish trust relationships locally for Kerberos principals king5.

On the first realm, enter:

```
TSS ADD(lscess1) KERBNAME(king5)
                 major_major_major
```

On the second realm, enter:

```
TSS ADD(lclint1) KERBNAME(major_major_major).
```

On the second realm, where king5 is foreign defines a foreign principal “king5” from the originating foreign realm MAJESTERIAL.CLIENT.COM and associates that principal locally with ACID “client1” in the local TSS.

```
TSS ADD(SDT) KERBLINK(kclint1)
             LINKNAME(‘/…/majesterial.client.com/king5’)
             KERBUSER(client1)
```

On the first realm where major_major_major is foreign, defines a foreign principal “major_major_major” originating from COLLOSSAL.SUCCESS.COM, and associates that principal locally with ACID “succes1” in the local TSS.

```
TSS ADD(SDT) KERBLINK(kscess1)
             LINKNAME(‘/…/collossal.success.com/major_major_major’)
             KERBUSER(succes1)
```

Chapter 7: z/OS Security Server Support

This section contains the following topics:

Disable RACF

RACF is IBM's SAF compliant security system. It is mainly concerned with system entry validation and resource permission.

RACF must be independently activated, and is not required to run the other Security Server components.

To disable RACF

1. Update the appropriate IFAPRDxx member and change the STATE field to:

   ```
   STATE(DISABLED)
   ```

2. Re-IPL the system.

   The changes take effect.

Examples: disabling RACF

This example disables the security server update the IFAPRDxx:

```
PRODUCT OWNER('IBM CORP')
NAME('z/OS')
FEATURENAME('Security Server')
ID(5647-A01)
VERSION(*)
RELEASE(*)
MOD(*)
STATE(DISABLED)
```

In this example IFAPRDxx is updated to disable the RACF component of the security server but continues use of the DCE component:

```
PRODUCT OWNER('IBM CORP')
NAME('z/OS')
FEATURENAME('Security Server')
ID(5647-A01)
VERSION(*)
RELEASE(*)
MOD(*)
STATE(ENABLED)

PRODUCT OWNER('IBM CORP')
NAME('z/OS')
FEATURENAME('RACF')
ID(5647-A01)
VERSION(*)
RELEASE(*)
MOD(*)
STATE(DISABLED)
```

DCE Security Server

The DCE Security Server is a separate security product that provides authentication services for users and servers running DCE applications.

In a DCE environment, one DCE Security Server must exist. A DCE security server provides a central hub for system entry validation and single-sign on authentication for all DCE platforms within a connected DCE environment. A key concept of a DCE security server is that it provides an independent or so-called third-party platform in which to validate and authenticate security requests.

How DCE Works

When a user passes from one DCE platform to another, the target platform passes information about the user (user credentials) along with other information to the DCE security server for authentication and authorization. The DCE security server authenticates such requests by checking the supplied user credentials against those stored in the DCE security server's security repository and/or security registry. In performing this authentication, the DCE security server follows an authentication algorithm, which involves not only the user credentials but also involves encryption keys known for each platform. The algorithm is standards-based and is platform independent. As a result, multiple vendors and platforms offer a DCE security server.

IBM's z/OS OE/DCE Security Server product allows an IBM mainframe to perform these functions.

Using CA Top Secret as a Repository

With z/OS O/E DCE security server, CA Top Secret acts as a repository for information needed to support the DCE authentication process. CA Top Secret holds DCE-specific userid information and encryption keys. A 'DCE segment' can be defined for any userid. The DCE userid segment allows six DCE-specific fields to be maintained for any userid. For example, one field named DCEKEY identifies each user's DCE password. During authentication, the DCE security server retrieves this and other information from CA Top Secret following formal SAF interfaces. This information is then used separately to authenticate and authorize the DCE security server request.

CA Top Secret provides support for:

- The DCE segment for any userid
- The KEYSMSTR resource class used to hold keys for DCE password encryption
- The DCEUUIDS resclass used to track the correspondence between z/OS userids and DCE UIDs
- Four new callable services used to exchange information between the ESM and the DCE security server

DCE Security Server Protection

To protect the DCE security server under CA Top Secret:

- Add a DCE segment allowing specification of six DCE-related fields to any CA Top Secret defined user. For example:

```
TSS ADD(acid) DCENAME(jordanm)
              DCEFLAGS(AUTOLOGIN)
              UUID(00000075-71db-21cf-b500-08005a470ba1)
              HOMEUUID(abbc323c-5ce2-11cf-a61e-08005a470ba1)
              HOMECELL(/.../cis_test1.cis.dog.com)
```

- CA Top Secret extends this support by installing the TSSDCE ISPFEDIT-macro which can reformat the output of the IBM DCE export utility, MVSEXPT, to TSS format. To perform this reformat, edit the MVSEXPT output data set using ISPF edit and enter the TSO command %TSSDCE. This immediately reformats the MVSEXPT output data set being edited into TSS command format.

For more information, see the IBM DCE documentation for the OE/DCE security server.

Firewall Technologies

z/OS provides the ability to run a firewall under USS. Support is distributed with the Communication Server and with the Security Server. The z/OS Firewall Technologies reduces, but does not necessarily eliminate the need for a non-z/OS platform firewall. The firewall is not configured using CA Top Secret. Administration is performed through configuration files.

To set up the z/OS Firewall Technologies with CA Top Secret

1. Enter the commands:

```
TSS CREATE (FWGRP) TYPE(GROUP)
           NAME('Firewall Group')
           DEPT(OMVSDEPT)

TSS ADD(FWGRP)
      GID(nn)
```

   A group definition for use with the firewall is created. Any unused GID number is allowed.

2. Enter the commands:

```
TSS CREATE(FWKERN) TYPE(USER)
                   NAME('Firewall Startup ID')
                   DEPT(OMVSDEPT)
                   FACILITY(STC,BATCH)
                   PASS(password,0)

TSS ADD(FWKERN) GROUP(FWGRP)
                DFLTGRP(FWGRP)
                HOME(/usr/lpp/fw/home/fwkern/)
                OMVSPGM(/bin/sh)
                UID(0)

TSS ADD(STC) PROCNAME(FWKERN)
             ACID(FWKERN)

TSS MODIFY(OMVSTABS)
```

   A Firewall startup address space ID is defined.

3. Enter the commands:

```
TSS ADD(anydept) IBMFAC(FWKERN.)

TSS PERMIT(FWKERN) IBMFAC(FWKERN.START.REQUEST)
                   ACCESS(UPDATE)
```

   FWKERN is allowed to issue start commands.

4. Enter the commands:

```
TSS ADD(STC) PROCNAME(ICAPSLOG)
             ACID(FWKERN)

TSS ADD(STC) PROCNAME(ICAPSOCK)
             ACID(FWKERN)

TSS ADD(STC) PROCNAME(ICAPPFTP)
             ACID(FWKERN)

TSS ADD(STC) PROCNAME(ICAPFLOG)
             ACID(FWKERN)

TSS ADD(STC) PROCNAME(ICAPTNAT)
             ACID(FWKERN)
```

   Additional started tasks used by the firewall daemons are defined.

5. Enter the commands:

```
TSS PERMIT(FWKERN) DSN(TCPIP.*)
                   ACCESS(READ)
```

   FWKERN is given access to READ the TCP/IP data sets.

   The high level qualifier of these data sets might have been renamed from “TCPIP” when installed on your system.

6. Enter the commands:

```
TSS PERMIT(FWKERN) IBMFAC(BPX.SMF)
                   ACCESS(READ)
```

   The FWKERN ACID is permitted to the SMF logging facility.

7. Enter the commands::

```
TSS PERMIT(FWKERN) IBMFAC(BPX.DAEMON)
                   ACCESS(READ)
```

   The PFTP server is permitted to the BPX.DAEMON facility.

8. Enter the command::

```
TSS ADD(administrator) GROUP(FWGRP)
```

   Firewall administrators are members of the group FWGRP or have superuser authority.

9. Enter the commands:

```
TSS ADD(dept) CSFSERV(service-name)

TSS PERMIT(acid) CSFSERV(service-name)
                 ACCESS(READ)
```

   Firewall Technologies has the ability to invoke z/OS Integrated Cryptographic facilities to perform internal security functions. These services are protected using the resource class CSFSERV. Users are now permitted to the individual services necessary.

For information on the individual service-names, see the IBM Firewall manuals and the ICSF/MVS Administrators Guide.

LDAP Server

IBM provides a Lightweight Directory Access Protocol (LDAP) Server with z/OS that uses a DB2-based file to store directory information such as email accounts.

To set up the z/OS LDAP Server with CA Top Secret

1. Enter the commands:

```
TSS CREATE (LDAPGRP) TYPE(GROUP)
                     NAME('LDAP Group')
                     DEPT(OMVSDEPT)
TSS ADD(LDAPGRP) GID(nn)
```

   A group definition for use with the LDAP Server is created. Any unused GID number is allowed.

2. Enter the commands:

```
TSS CREATE(LDAPSRV) TYPE(USER)
                    NAME('LDAP Startup ID')
                    DEPT(OMVSDEPT)
                    FACILITY(STC,BATCH)
                    PASS(password,0)

TSS ADD(LDAPSRV) GROUP(LDAPGRP)
                 DFLTGRP(LDAPGRP)
                 HOME(/)
                 OMVSPGM(/bin/sh)
                 UID(0)

TSS ADD(STC) PROCNAME(LDAPSRV)
             ACID(LDAPSRV)
TSS MODIFY(OMVSTABS)
The LDAP Server startup address space identifier is defined.
```

3. Enter the commands:

```
TSS ADD(anydept)   IBMFAC(BPX.)

TSS PERMIT(LDAPSRV) IBMFAC(BPX.DAEMON)
                    ACCESS(READ)

TSS PERMIT(LDAPSRV) IBMFAC(BPX.SERVER)
                    ACCESS(UPDATE)
```

   The ACID for the LDAP server started task is granted access to the IBMFAC resources.

Integrated Cryptographic Services

The IBM high powered cryptographic coprocessor allows z/OS applications to exploit cryptography. The z/OS Security Server provides API's to invoke these cryptographic services (ICSF). Various functions involved with the management of keys are provided in this service. These services combine to manage public keys.

CA Top Secret provides the following resource classes to allow ICSF to be secured and audited:

CSFKEYS

This class secures encryption keys. The value, which is owned and permitted, is the key *label.* The key label is in the CKDS or PKDS when a key is defined.

If CFSKEYS is defined to support masking and you want to allow ALL access, use CFSKEYS(**). This allows the correct entity to be picked up. If this is not done and an audit record is cut the resource name is blank.

CSFSERV

This class secures the various cryptographic services, needed to encrypt data and manage keys. For information, see the *Integrated Cryptographic Service Facility: Administrator's Guide*.

Note: If the certificate's private key resides in an ICSF storage facility and the format of PKCS12DER or PKCS12B64 is specified in the TSS EXPORT command, the command is rejected. To use ICSF, you must have cryptographic hardware installed and enabled on your system. ICSF is the interface to the Cryptographic hardware on z/OS.

Chapter 8: Controlling Access to the Hierarchical File System

This section contains the following topics:

HFS Control Using the Native UNIX Security Model

The HFS is a tree-structured file system consisting of directories and files. It uses the forward slash (/).

Each file and directory is assigned an owning UID and an owning GID. The assignment is defined and saved in the file system, not in the external security product.

There are two methods to secure a HFS:

- Internal USS based on the UNIX model of security
- External security that uses standard CA Top Secret security authorizations

Important! These methods are mutually exclusive.

User Categories and Access Levels

Three categories of users can access each directory and file in the HFS:

- The file owner
- The group that owns the file
- All other users defined to USS

Different access levels can be set for any of these three categories. For example, permissions can be defined so that the file owner gets READ and WRITE access, a member of the file's group gets only READ access, and all other users have neither READ nor WRITE access to the file.

HFSSEC and HFSACL Control Options

Native UNIX security assigns a user ID and a group ID as file owner. Ownerships are assigned in UNIX file creation or when updated by UNIX commands chown, chgrp, chmod, and setfacl.

HFSSEC(ON)

CA SAF security for USS is in effect. Security for files in HFS is determined by resources and permissions in the HFSSEC resource class. File and group ownership in native UNIX security is ignored.

HFSSEC(OFF)

An OMVS user attempting to access a file has their UID evaluated against the UID assigned as the owner of the file. If the UIDs match, access is allowed according to the user bit security setting assigned to the file. If the UIDs do not match, native UNIX security proceeds to group security. If the file's ownership GID matches the GID of any GROUP added to the accessing user, access is granted.

HFSACL(OFF)

Access control lists beyond the base level access list "u::uflags,g::gflags,o::oflags" is ignored. Access control lists are assigned by the setfacl OMVS command. They are used to assign access to a file by more than a single UID owner, or more than a single GID owner.

HFSSEC(NO) and HFSACL(ON)

The accessing user's UID is matched against the UID in every access control list assigned to the file. If matched the user-bit settings of the matching ACL determine user access to the file. If the accessing user does not match any of the user ACL settings, the accessing user's assigned list of groups is checked for a match against the group assigned to the file's access control lists. If an ACL matches by group, that assignment of group-bit settings is used.

Unmatched IDs

If no ACL user or group assigned to the file matches the accessing user's UID or the assigned list of GID's, native UNIX security checks if the user ID has the RSTDACC attribute and the READ access to permission for UNIXPRIV(RESTRICTED.FILESYS.ACCESS) to determine whether the file's other access bit settings are used. The following table details various unmatched ID scenarios:

User is "restricted"	READ access to UNIXPRIV(RESTRICTED.FILESYS.ACCESS)	Result
Yes	Yes	Check "other" bits to determine access.

Yes	No	Bypass check of "other" bits and deny access.
No	Yes	Check "other" bits to determine access.
No	No	Check "other" bits to determine access.

If "other" file access bits are not to be used, access is rejected.

If "other" file access bits are to be used, access is allowed only if the operation attempted is compatible with the bit setting.

Group Access Checks

When group access checks are performed, CA Top Secret compares the GID of the file to the GID of the default group defined in DFLTGRP. If these do not match, CA Top Secret compares the file's GID to the GIDs of all of the groups defined to the ACID in the GROUP field.

To add GIDs to the GROUP, enter:

```
TSS ADD('acidname') GROUP('group name')
```

If a match is found, CA Top Secret uses group permissions to determine the user's access to the file.

Obtain Security Information for a File within OpenEdition

To obtain security information for a file within OpenEdition, perform one of the following actions:

- Use the OpenMVS ISPF Shell.
- Enter the ls -l or ls -E shell command.
- Run a stat( ) or fstat( ) function in a program.

The system returns the following information:

- Type of file or directory
- Owner permission bits
- Group permission bits
- Other permission bits

If the system can locate the TSO/E user ID and CA Top Secret group name that correspond to the file's UID and GID permission bits, the system displays the ID and group name. If the UID is 0, the system displays the user that owns the UID. For CA Top Secret, the owner is the MSCA ACID, in which case the display shows *MSCA* (not the MSCA ACID name).

If the system cannot find the corresponding TSO/E user ID and CA Top Secret group name, the system displays the UID and GID.

OpenEdition Commands

Open edition shell commands that change file/directory authorizations include:

CHMOD

Change permission bits for a file/directory.

CHOWN

Change the owner or group for a file or directory.

CHGRP

Change the group of a file.

For information on the Hierarchical File System and setting file permissions, see the IBM *OpenEdition User's Guide* and *OpenEdition Planning.*

Processes that Affect HFS Security

When using the UNIX security model, various options can affect the file validation process. The processes and their effect on file security or validation are described in this section.

HFS FASTPATH Checking

OMVS issues a SAF call at initialization that checks if access is authorized for OMVS to the FACILITY class resource BPX.SAFFASTPATH. When the IBMFAC(BPX.SAFFASTPATH) resource is defined (for example, TSS ADD(dept) IBMFAC(BPX.)), OMVS performs permission bit checking internally instead of calling the external security manager bypassing any audit trail of violations. This internal checking is FASTAUTH processing.

To activate FASTPATH checking for all users, enter:

```
TSS ADDTO(anydept) IBMFAC(BPX.SAFF)
```

To deactivate FASTPATH checking for all users, enter:

```
TSS REMOVE(anydept) IBMFAC(BPX.SAFF)
```

Do not give the STC ACID associated with the OMVS started task the NORESCHK attribute.

MOUNT NOSECURITY

You have the option to MOUNT a file system or part of a file system with or without SECURITY. The use of the MOUNT command requires superuser authority. If the file system is mounted with the NOSECURITY option, USS makes access checks against system credentials (superuser) rather that against user credentials. Access is allowed.

Program Control in the UNIX Environment

When the BPX.DAEMON and BPX.SERVER facilities are active, processing authorized functions, such as SETUID, requires that programs or executables be loaded from an authorized library. In an CA Top Secret environment, these authorized data sets are any library in the LPA list, the APF list, or LINKLIST. If a program is loaded from the HFS or an MVS data set not on the approved lists, the TCBNCTL flag, referred to as the “dirty bit,” is set. This results in authorized functions failing if attempted in the “dirty” environment.

When an executable or program is requested in an OMVS environment, OMVS finds the executable in the HFS and loads from there unless the program controlled extended attribute, or “sticky bit,” is set. If this sticky bit is set on the executable file, OMVS uses normal MVS load processing. To avoid the dirty bit being set the executables in the HFS set the sticky bit on using the chmod command.

HFS Control Using CA SAF HFS Security

z/OS brings the MVS and UNIX operating systems together onto one hardware platform. Although some interoperability between MVS and UNIX exist, each environment retains its own distinct data structures and methods of access control.

UNIX data is kept in a Hierarchical File System (HFS). From the UNIX perspective, the HFS contains many discrete data files. From the MVS perspective, the HFS is one data set and can only be controlled as one data set. MVS can control access to the entire file system, but not to the individual files within the HFS.

HFS files are protected by file permission bit settings set when the file owner creates the file. A superuser, a user privilege that grants much more authority than just security administration, can only perform centralized administration. MVS resources are protected by resource access that can be set up in advance by scoped security administrators.

CA SAF HFS security provides single-point security access control, administration, and reporting for both MVS and UNIX resources. CAIENF services present access events to CA Top Secret for validation. Administrators use familiar commands and rules to protect UNIX files and functions, restricting access based on the CA Top Secret ACID permissions instead of the UNIX UID or GID numbers. HFS access loggings and violations are reported in the standard CA Top Secret reports.

CA SAF File Access Security

When using CA SAF HFS security, native file permission bit security is bypassed. File access is validated by CA Top Secret through resource permissions. All the benefits of resource permissions can be utilized, including masking, profiles, scoping, and reporting.

Path Name Manipulation

A path name can be up to 1023 characters in length, except when used in the JCL PATH= keyword where the limit is 255 characters. The path name is case sensitive and can contain special characters. In order to allow external security to validate HFS files, CA Top Secret validation of HFS files requires path name manipulation.

Before validation, all path names are converted to upper case and, if necessary, truncated at 255 characters. An exit point is provided when file names reside in paths greater than 255 characters. The installation can use the exit to provide a meaningful name.

Path Name Translation

CA Top Secret resource authorization processing considers the period character (**.**) as a delimiter. This delimiter is used when permitting masked resources, such as, when providing security for data sets.

Path names use the slash character as a delimiter. Before a file is validated, the path name has all slash characters (except the first) translated into a period. Other special characters are translated into the dollar sign ($). These include characters that are used as masking characters in resource permissions. If not translated, these characters could create undesired results. The special characters include the period, asterisk, dash, plus, blank, and quote. An exit point is provided which can further modify any character to meet special needs, with the exception of the slash character which is always translated to a period delimiter.

The following table shows examples of path name translation:

Original path name	Translated path name	Sample resource authorizations	Security action
/bin/su	/BIN.SU	TSS PER(USER01) HFSSEC(/BIN.SU) ACCESS(NONE)	No access to switch user command
/u/user01/proj1/ file1.txt	/U.USER01.PROJ1.FILE1$TXT	TSS PERMIT(USER01) HFSSEC(/U.%.PROJ1.FILE1$TXT) ACCESS(ALL)	All access allowed
/usr/sbin/mknod	/USR.SBIN.MKNOD	TSS PER(SYSPROG) HFSSEC(/USR.SBIN.MKNOD) ACCESS(ALL)	Allow system programmers to create special characters

HFSSEC Resource Class

CA SAF HFS security resource class HFSSEC defines file and directory level access in CA Top Secret.

Permission Considerations

This section describes special considerations to be taken into account when administering permits for HFS resources.

In addition to access to HFS files, users also need access to directories. A user requires READ access to a directory in order to list the contents of that directory. When writing a permission, distinguish a directory permit from a file permit by not using a trailing period in the HFSSEC authorization.

Example: HFS permission

This example allows users to read the /BIN directory but only allows EXECUTE access to the files contained within the directory:

```
TSS PER(ALL) HFSSEC(/BIN)
             ACCESS(READ)

TSS PER(ALL) HFSSEC(/BIN.)
             ACCESS(EXEC)
```

Root Directory Permissions

The root directory is defined by the single character (/). With CA Top Secret the root directory must be owned using the special name ROOT. Files contained in the root directory (/) must be specified as the slash (/) followed by the file name. Therefore, the only valid permit for the ROOT directory (/) is that which allows read access to the directory itself.

Example: root directory permission

This example allows read and write access to file rootfile:

```
TSS PER(ALL) HFSSEC(ROOT)
             ACCESS(READ)

TSS PER(ALL) HFSSEC(/ROOTFILE)
             ACCESS(UPDATE)
```

ACCESS Keywords

Permits administered to secure HFS file resources should specify the ACCESS keyword to identify the type of access to the file. If the access keyword is not used, READ access is implied. The access keywords are:

ALL

Allows all of the following.

ALTER

Allows create and delete access to a file.

CONTROL

A special access not used for normal file access validation. This is used with HFS function security to allow a user to change file attributes.

EXECUTE

Allows execute access to a file, usually a program file.

READ

Allows read access to a file.

UPDATE

Allows write access to a file.

Reporting

Audit records created by HFS file access checks, violations, and audit events are written to the Audit Tracking File and accessed by the TSSUTIL report utility. TSSUTIL integrates these events among other events according to the report generation criteria.

Secure HFS Functions

In addition to file access security, HFS functions can also be secured. These functions can be a system action, such as setting a ptrace or a job's priority, or they can be file-related, such as changing the file mode or audit settings.

A system function is secured by a rule in the IBMFAC class, while a file-related function is secured by a combination of an IBMFAC class rule and a HFS file resource rule. By following this approach, changes to file attributes can be permitted at a global basis, or restricted to a particular file.

The resource name format for HFS IBMFAC rules is: .BPX.CAHFS.function

Example: HFS function security

This example secures HFS functions:

```
TSS PER(USER01) IBMFAC(BPX.CAHFS.function)
                ACCESS(READ)
```

System Functions

To perform a system function, the user requires READ access to the corresponding IBMFAC:

BPX.CAHFS.CHANGE.PRIORITY

Allows a user to change the scheduling priority of a process, process group, or user.

BPX.CAHFS.SET.PRIORITY

Allows a user to set the scheduling priority of a process, process group, or user.

BPX.CAHFS.SET.RLIMIT

Allows a user to set the resource limit for the calling process.

BPX.CAHFS.MOUNT

Allows a user to mount file systems. For z/os 1.13 and above, an additional check for the BPX.CAHFS.USERMOUNT resource occurs if the check for BPX.CAHFS.MOUNT fails.

BPX.CAHFS.USERMOUNT

Allows a nonprivileged user to mount a file system. If the check for BPX.CAHFS.MOUNT resource fails, BPX.CAHFS.USERMOUNT is checked. BPX.CAHFS.USERMOUNT differs from BPX.CAHFS.MOUNT support because with UNIX System Services extra criteria are required for the user to be able to mount the file system. These criteria include needing to mount on an empty directory, needing permission to the mount point directory and the file system being mounted, and others. For a complete list of these criteria, see IBM's *z/OS UNIX System Services Planning*.

Because the user must own or have permission to the file system being mounted and the mount point directory, the following permits are required:

- ALTER access to the HFSSEC class where the resource name represents the file system data set
- ALTER access to the HFSSEC class where the resource name represents the mount point directory.

Example:

```
mount –t zfs –f OMVS.HFS.BOBFILE –a unmount –s nosetuid /user/bob
```

For user BOB to mount OMVS.HFS.BOBFILE at the mount point /user/bob using the above OMVS shell command, the following three permits are required:

```
TSS PERMIT(BOB) IBMFAC(BPX.CAHFS.USERMOUNT) ACCESS(READ)
TSS PERMIT(BOB) HFSSEC(OMVS.HFS.BOBFILE) ACCESS(ALTER)
TSS PERMIT(BOB) HFSSEC(/USER.BOB) ACCESS(ALTER)
```

BPX.CAHFS.USERUNMOUNT

Allows a nonprivileged user to unmount a file system. Otherwise, for UNIX System Services, nonprivileged users cannot unmount a file system that they did not mount.

BPX.CAHFS.PTRACE

Allows a user to control and debug another process. Although the user need not be defined as a superuser to use this function, access to this resource does not give the user more authority than a superuser would have. If the user attempts to debug a program running with SETUID or SETGID (that is, a program that switches user identification), access to the function is denied.

BPX.CAHFS.CREATE.LINK

Allows a user to create a hard link to an existing file. A hard link is essentially another name for the same file data. If the original file is removed, the hard link still points to the file data. The data is not deleted until the last link is removed. The user requires a permission with ACCESS(ALTER) to the HFS file resource for both the original file and the link file. When data associated with a hard link is accessed, the CA-ENF/USS service requests the file name from USS. Regardless of the actual path accessed, the file name that is returned may be the hard link name or the original file name. Therefore, when a hard link exists, you must maintain permissions for both the link name and the original name.

BPX.CAHFS.CREATE.EXTERNAL.LINK

Allows a user to create an external link to an object outside of the file system, such as an MVS data set. An external link is a file that contains the name of an external object. If the external object is removed, the external link still contains the name of the nonexistent object.

BPX.CAHFS.CREATE.SYMBOLIC.LINK

Allows a user to create a symbolic link to an existing file. A symbolic link is a file that contains the name of another file. If the original file is removed, the file data is deleted but the symbolic link still contains a pointer to the nonexistent file. Symbolic link names are validated when the link is created and deleted. All other accesses are validated with the original file name. In addition to this resource, the user must also have a PERMIT with ACCESS(ALTER) to the HFS file resource for both the original file and the link file.

File Functions

File-related functions can be secured to various levels of granularity by determining a user's highest level of access to an IBMFAC resource. The ACCESS keyword of the IBMFAC resource authorization is used for this purpose. The following actions are taken based upon the ACCESS value:

ALL

The user is allowed to perform the function against all files.

NONE

The user is not allowed to perform the function against any files.

READ

(Or any access containing READ such as CONTROL or UPDATE.)

The user is allowed to perform the function if the user also has ACCESS(CONTROL) access to the HFS file.

The access level of CONTROL is not used in normal file access. It is utilized here to provide additional controls for file functions.

READ may also allow the function if the HFS file exists in the user's 'user path directory'. That is, if the file exists in the users directory matching the userid making the request. Normally the directory is chained off the /u directory but this can be altered by the user exit.

Because the absence of the ACCESS keyword in a permission implies READ access, specify ACCESS in all of the file function IBMFAC permissions so that you do not inadvertently allow greater access to functions than you intended.

HFS file permission settings and UID/GID ownership are not used for validation purposes when CA SAF HFS security is active.

File Functions (IBMFAC)

The following file functions are authorized via the IBMFAC ATTRIBUTE:

BPX.CAHFS.CHANGE.FILE.ATTRIBUTES

Allows a user to change extended file attributes, such as APF authorization and program control. Native z/OS UNIX services will issue an IBMFAC resource call to determine authorization to set the specific attribute, but not to specific files. Use of this file function resource provides additional control down to the file level.

BPX.CAHFS.CHANGE.FILE.AUDIT.FLAGS

HFS files contain two sets of audit flags, one that can be set by a normal user and the other that can only be set by an auditor. This resource allows a user to change user-audit flags in a file.

BPX.CAHFS.CHANGE.FILE.FORMAT

Allows a user to change the format of a file. Changes include defining text data delimiters or binary file format.

BPX.CAHFS.CHANGE.FILE.MODE

Allows a user to change any file mode information. This includes changes to file permission settings, setting the execution UID or GID indicators, and setting the “sticky” bit. Native z/OS UNIX permission settings are used for validation purposes only when CA SAF HFS security is inactive.

BPX.CAHFS.CHANGE.FILE.MODE.STICKY

Allows a user to set the “sticky” bit in the file mode information. The “sticky” bit causes a program to be loaded from MVS libraries instead of the HFS.

BPX.CAHFS.CHANGE.FILE.MODE.EUID

Allows a user to set the execution-UID indicator in the file mode information. When this indicator is set, the program runs under the UNIX UID of the file owner instead of the UID of the user running the program.

BPX.CAHFS.CHANGE.FILE.MODE.EGID

Allows a user to set the execution-GID indicator in the file mode information. When this indicator is set, the program runs under the UNIX GID of the file owner instead of the GID of the user running the program.

BPX.CAHFS. CHANGE.FILE.OWNER

Allows a user to change file owner UID setting. Native z/OS UNIX ownership settings are used for validation purposes only when CA SAF HFS security is inactive.

BPX.CAHFS. CHANGE.FILE.GROUP

Allows a user to change file owner GID setting. Native z/OS UNIX ownership settings are used for validation purposes only when CA SAF HFS security is inactive.

BPX.CAHFS. CHANGE.FILE.TIME

Allows a user to change the last access or modification time to the current time or a user-specified time. If the current time is to be set and the user has write access to the file, the function is allowed. If the user does not have write access or a user-specified time is to be set, access must be allowed to this IBMFAC resource.

Example: IBMFAC permissions

This example allows Thelma to change the file mode and owner for all files. Louise is allowed to change the file mode for only those files that reside in a certain directory, but is not allowed to change the file owner in any file:

```
TSS PER(THELMA) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE)
                ACCESS(ALL)

TSS PER(LOUISE) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE)
                ACCESS(CONTROL)

TSS PER(THELMA) IBMFAC(BPX.CAHFS.CHANGE.FILE.OWNER)
                ACCESS(ALL)

TSS PER(LOUISE) HFSSEC(/certain.directory.)
                ACCESS(ALL)
```

CA SAF HFS Security

CA SAF HFS security is an application of CAIENF/USS. This security application is activated when the appropriate DCM modules are linked into the ENF database.

The process to implement CA SAF HFS security is:

- Determine if exit processing is required for path name translation, user path definition, or to enable file ownership.
- If using the exit, assemble and link the exit code using the sample SMPE usermod in OPMAT member UD00001.
- Define HFS file and function resource authorizations.

  CA recommends that all the function resources described in the previous sections be defined. Use the CA SAF HFS ADD/PERMIT Generation Utility to assist in creating these resource rules.
- If you utilize the user file ownership feature of CA SAF HFS security define authorizations for users.
- Verify that the proper level of CAIENF is available to support ENF/USS.

  CA Common Services for z/OS with the following APARs provides this support: LO89578 through LO89581, LO89584, LO92642, and LO94652, and LO94657.
- The ENF started task must be a valid OMVS user. Ensure the ENF ACID specifies a group. Install the following DCM modules into the ENF database using the ENFDB utility program: CARRDCM0 (Framework) and J163DCM0(CA Top Secret).
- Defining a VLF class for use as a cache can enhance performance of ENF/USS. The cache size is determined by the MAXVIRT specification. The number of cache entries is approximated by dividing the defined amount of VLF storage by the average size of your path names. Add the following to your current COFVLFxx member in SYS1.PARMLIB:

  ```
  CLASS NAME(CAENFU)      /* ENF/USS pathname cache */
        EMAJ(PATHCACHE)   /* Major name             */
        MAXVIRT(256)      /* 1 megabyte             */
  ```

  Adding the NODSNCHK attribute to the BPXOINIT logonid during initial testing allows OMVS to successfully initialize without violations. Once appropriate authorizations are in place, the NODSNCHK attribute should be removed.

  The following message is issued by CAIENF/USS at ENF startup when CA SAF HFS security is successfully initialized:

  ```
  CARR036I - SAFHFINT / J163 Now Initialized
  ```
- Set the CA Top Secret Control Option HFSSEC to ON: HFSSEC(ON)
- Run TSSUTIL during the implementation phase. Review violations and loggings for HFSSEC and IBMFAC resource types and create appropriate authorizations.

HFSSEC Control Option

The CA Top Secret Control Option turns HFS security ON and OFF. This control option is set in the TSS PARMS startup member or can be dynamically set via a TSS MODIFY command.

To determine the current active setting of HFSSEC, enter:

```
TSS MODIFY STATUS(BASE)
```

HFSSEC(ON)

Enables CA SAF HFS security. Normal z/OS UNIX security access validation is bypassed. This includes checking of file permission bits, superuser status and normal z/OS UNIX security services.

HFSSEC(OFF)

Disables CA SAF HFS security. Normal z/OS UNIX security access validation is enabled. This includes checking of file permission bits, superuser status and normal z/OS UNIX security services.

Exit Processing

An exit point is provided for installation-specific processing. This exit is called for both an initialization function, where options involving pathname translation and user path processing can be selected, and a pathname translation function, where final modification to the pathname can be made before validation is performed.

The exit must be reentrant and capable of running AMODE(31) and RMODE(ANY). The exit name is SAFHFUSR and must be linked together with load module SAFHFSEC. A sample SMPE usermod to perform this is in CAIJCL member UD00001.

Upon entry to the exit, register R1 points to a list of addresses. The end of the list is indicated by the high order bit in the last full word.

Initialization Parameter Addresses

For an initialization function, the exit is passed the following parameter addresses:

+0

The address of a single byte containing the character 'I' indicating that this is an initialization function.

+4

The address of a 512-byte work area for the use of the exit program.

+8

The address of a 255-byte field in which the user can return the path location where user directories are located. Upon input, this field contains hex zeros.

+12

The address of a single byte which, when set to 'Y' by the exit, indicates that user ownership of files is in effect.

+16

The address of a 256-byte translation table, which is used to translate certain special characters in a path name.

Exit Return Processing

When the exit returns a user directory path location, CA SAF HFS processing uses that path name to determine if the path name to be validated should be translated to a form such that the user ID of the owner of the path becomes the high-level qualifier of the path name. This allows HFS file rules to be written at the user level. The default is that no translation takes place for user directories.

If the exit returns the value **/**u**/** as the user directory path name location, and the file accessed is /u/user01/xfile, then the resource name validated is $$USER01.XFILE.When the exit returns the character 'Y' indicating that user ownership of files within one's own directory is in effect, no validation is performed when the current user's logonid matches that in the user directory.

This option is meaningless if a user directory path location is not also returned.

Example: access validation

In this example validation is bypassed when USER01 accesses file /u/user01/xfile:

```
TSS PER(USER01) HFSSEC($$%)
                ACCESS(UPDATE)
```

Character Translation Table

The supplied translate table is in a format acceptable as input to the assembler TR instruction. The default translate table translates all slash characters in a path name, with the exception of the leading slash, to a period character. Other special characters are translated into the dollar sign ($). These include characters that are used as masking characters in resource rules. If not translated, these characters could create undesired results. The special characters include the period, asterisk, dash, plus, blank, and quote.

The exit point can further modify any character in the table to meet special needs (with the exception of the slash character).

Path Name Translation Parameters

For a path name translation function, the exit is passed the following parameter addresses:

+0

The address of a single byte containing the character ‘P’ indicating that this is a path name translation function.

+4

The address of a 512-byte work area for the use of the exit program.

+8

The address of a 255-byte field containing the resource name as modified by CA SAF HFS processing. This is the name that is used for validation. The exit can return a modified path name in this same field.

+12

The address of a 1023-byte field containing the original unmodified path name.

The exit can use this exit function to make any specific modifications to the path name beyond that already performed by CA SAF HFS security processing.

CA SAF HFS ADD/PERMIT Generation Utility

A set of utility programs is provided to generate CA Top Secret commands used as a starter set of resource definitions and authorizations for new implementations. The HFS resource authorizations created give access based upon the file permission bits defined for groups and 'other' users. The rules give users the same default access to files they have when not running CA SAF HFS security. The generated authorizations must be modified to allow appropriate users greater access to the directory resources than that granted to the general user community. Because of the number of files that can be contained in the HFS the utility only covers the directories. File permissions can be added before running the REXX exec.

The HFS File System contains directories and files in a tree structure. In order to quickly add CA Top Secret security a set of procedures and utilities are provided.

File Protection Procedure

Use this procedure to protect your current file system:

1. Run the OMVS "**ls** -**lRA**" command in a batch TMP. Direct the output to a standard DASD file. This file must be allocated with RECFM=VB.

   Issue the **ls** command from the OMVS shell, directing the output to a HFS file. The options **-lRA** must be specified (the character following the dash is a lower case letter 'L', not the number one). The file can then be copied into a MVS data set using the OGET command. For example:

```
ls  -lRA  /  >>directory_information_file
OGET '/directory_information_file' 'mvs.input.file'
The resulting file data should look similar to this:
/:
total 232
drwx------   3 USER     OPENMVS        0 Jun  3  1998 JavaS390
drwxr-xr-x   4 USER                     0 May  7  1998 bin
drwx--x--x   2 USER     OPENMVS        0 Oct  1  1997 dev
drwxr-xr-x   8 USER     OPENMVS        0 Nov  4 17:05 etc
drwxr-xr-x   2 USER                     0 Jan 20  1998 lib
drwxrwxrwx   2 USER                     0 Jan 19 11:51 tmp
drwxr-xr-x   8 USER     OPENMVS        0 Jan 15 15:47 u
drwxr-xr-x  11 USER                     0 Jan 20  1998 usr

/JavaS390:
total 16
drwxrwxrwx   7 USER     ZEROGRP        0 Sep 25  1997 J1.1.1
```

2. Run HFSPASS1.

   This job reads the file from the previous step, creates and intermediate data set and then sort that data creating a file for the next step.

```
//        JOB
//STEP1   EXEC PGM=HFSUTIL1,REGION=0M
//SYSABEND DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//HFSINPUT DD DSN=????.????.????,DISP=SHR
//EXTRACT  DD DSN= SORT.INPUT,UNIT=3390,
//        DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(15,1),RLSE),
//        DCB=(RECFM=FB,LRECL=300,BLKSIZE=6000)
/*
//STEP2   EXEC PGM=SORT,REGION=0M
//SYSOUT  DD SYSOUT=*
//SORTWK01  DD UNIT=3390,SPACE=(CYL,5)
//SORTWK02  DD UNIT=3390,SPACE=(CYL,5)
//SORTWK03  DD UNIT=3390,SPACE=(CYL,5)
//SORTWK04  DD UNIT=3390,SPACE=(CYL,5)
//SORTIN    DD DSN=SORT.INPUT,DISP=(OLD,DELETE,KEEP)
//SORTOUT   DD DSN=SORT.OUTPUT,UNIT=3390,
//         DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(15,1),RLSE),
//         DCB=(RECFM=FB,LRECL=300,BLKSIZE=6000),VOL=SER=SCAC16
//SYSIN     DD *
 SORT FIELDS=(1,264,CH,A)
/*
```

3. (Optional) Alternatively, the input file from the first step can point directly to the directory information file created from the **ls** command.

   If using this format, the LRECL value specified in the JCL must be at least as large as the largest record in the file. The BLKSIZE value should be a value at least 8 greater than the LRECL. The PATH name must be the full path name of the file containing the directory information. A sample statement follows:

```
//HFSINPUT  DD   PATH='/directory_information_file',
//               PATHOPTS=(ORDONLY),FILEDATA=TEXT,
//               RECFM=VB,LRECL=nnn,BLKSIZE=nnn
```

4. Edit the file created in step 2:

   a. At the beginning of the data set are records to build a /group profile cross-reference table. The formats of those records are:

   AAAAAAAA - xxxxxxxx

   where AAAAAAAA is the name of an OMVS group

   Change xxxxxxxx to a profile to be used for any permissions needed by this group. In our example OPENMVS is the group and you must assign a profile name to "xxxxxxxx".

   This is not a complete list of all groups, only those ACIDs that needs a specific permission given.

   b. After those records are several TSS ADD or TSS ADDTO commands. These are all of the ownership's that are required for the conversion to meet with success. In these statements the xxxxxxxx (ACID name) needs to be modified to whatever ACID the client wants to own the specified resources.

```
OPENMVS  - xxxxxxxx
TSS ADD(xxxxxxxx) HFSSEC(ROOT)
TSS ADD(xxxxxxxx) IBMFAC(BPX.CAHF)
TSS ADD(xxxxxxxx) HFSSEC(/bin)
TSS ADD(xxxxxxxx) HFSSEC(/dev)
TSS ADD(xxxxxxxx) HFSSEC(/etc)
TSS ADD(xxxxxxxx) HFSSEC(/lib)
TSS ADD(xxxxxxxx) HFSSEC(/opt)
TSS ADD(xxxxxxxx) HFSSEC(/samples)
TSS ADD(xxxxxxxx) HFSSEC(/tmp)
TSS ADD(xxxxxxxx) HFSSEC(/u)
TSS ADD(xxxxxxxx) HFSSEC(/usr)
TSS ADD(xxxxxxxx) HFSSEC(/JavaS390)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.ATTRIBUTES)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.AUDIT.FLAGS)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.FORMAT)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.GROUP)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE.EGID)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE.EUID)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE.STICKY)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.OWNER)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.TIME)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.PRIORITY)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CREATE.EXTERNAL.LINK)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CREATE.LINK)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.CREATE.SYMBOLIC.LINK)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.MOUNT)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.PTRACE)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.SET.PRIORITRY)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.SET.RLIMIT)
TSS PER(ALL) FOR(14) IBMFAC(BPX.CAHFS.UNMOUNT)
```

```
TSS PER(ALL) HFSSEC(ROOT) ACCESS(READ)
ALL     //bin
ALL     //dev
ALL     //etc
ALL     //lib
ALL     //opt
```

5. Run HFSPASS2.

   HFSPASS2 reads the edited data set and produces a data set containing all the TSS commands to be executed. For example:

```
//        JOB
//STEP3   EXEC PGM=HFSUTIL2,REGION=0M                              //SYSABEND DD
SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//EXTRACT  DD DSN=SORT.OUTPUT,DISP=SHR
//PRMOUT   DD DSN=TSS.CMDS,UNIT=3390,VOL=SER=SCAC16,
//         DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(15,1),RLSE),
//         DCB=(RECFM=FB,LRECL=300,BLKSIZE=6000)
Example 2.1  Output from the HFSUTIL2
TSS ADD(xxxxxxxx) HFSSEC(ROOT)
TSS ADD(xxxxxxxx) IBMFAC(BPX.CAHF)
TSS ADDTO(xxxxxxxx) HFSSEC(/bin)
TSS ADDTO(xxxxxxxx) HFSSEC(/dev)
TSS ADDTO(xxxxxxxx) HFSSEC(/etc)
TSS ADDTO(xxxxxxxx) HFSSEC(/lib)
TSS ADDTO(xxxxxxxx) HFSSEC(/opt)
TSS ADDTO(xxxxxxxx) HFSSEC(/samples)
TSS ADDTO(xxxxxxxx) HFSSEC(/tmp)
TSS ADDTO(xxxxxxxx) HFSSEC(/u)
TSS ADDTO(xxxxxxxx) HFSSEC(/usr)
TSS ADDTO(xxxxxxxx) HFSSEC(/JavaS390)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.ATTRIBUTES)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.AUDIT.FLAGS)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.FORMAT)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.GROUP)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE.EGID)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE.EUID)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE.STICKY)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.MODE)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.OWNER)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.FILE.TIME)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CHANGE.PRIORITY)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CREATE.EXTERNAL.LINK)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CREATE.LINK)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.CREATE.SYMBOLIC.LINK)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.MOUNT)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.PTRACE)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.SET.PRIORITRY)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.SET.RLIMIT)
TSS PERM(ALL) FOR(14) IBMFAC(BPX.CAHFS.UNMOUNT)
TSS PERM(ALL) HFSSEC(ROOT) ACCESS(READ)
TSS PERMIT(ALL) HFSSEC(/bin) ACCESS(READ,EXEC)
TSS PERMIT(ALL) HFSSEC(/dev) ACCESS(EXEC)
TSS PERMIT(ALL) HFSSEC(/etc) ACCESS(READ,EXEC)
TSS PERMIT(ALL) HFSSEC(/lib) ACCESS(READ,EXEC)
```

```
TSS PERMIT(ALL) HFSSEC(/opt) ACCESS(READ,EXEC)
TSS PERMIT(ALL) HFSSEC(/samples) ACCESS(READ,EXEC)
```

6. Run REXX exec to execute the commands in the data set.

```
/*   REXX   */
/*
     This EXEC will have as input a dataset name and will read
     that dataset and issue the TSS commands in it.
*/
  arg dsn .                  /*  get data set name                */
  if dsn = '' then dsn = '????.?????.????'
  'ALLOC FI(PERMIN) DS(''dsn'') SHR REUSE VOL(??????)'
  eof = 'no'
  do while eof = 'no'
    'execio 1 diskr PERMIN'
    if rc = 2
       then do
          eof = 'yes'
          end
       else do
          pull record
          say 'Record is: ' record
          record
          end
  end
    'execio 1 diskr permin ( finis'
    'FREE FI(PERMIN)'
  exit 0
```

CA SAF HFS ADD/PERMIT Generation Utility

A set of utility programs is provided to generate CA Top Secret commands used as a starter set of resource definitions and authorizations for new implementations. The HFS resource authorizations created give access based upon the file permission bits defined for groups and 'other' users. The rules give users the same default access to files they have when not running CA SAF HFS security. The generated authorizations must be modified to allow appropriate users greater access to the directory resources than that granted to the general user community. Because of the number of files that can be contained in the HFS the utility only covers the directories. File permissions can be added before running the REXX exec.

The HFS File System contains directories and files in a tree structure. In order to quickly add CA Top Secret security a set of procedures and utilities are provided.

Chapter 9: Resource Validation and Auditing

This section contains the following topics:

XPARMS (Major)

ENF automatically calls CA Top Secret (ENFCICS is active) when access is attempted for any CICS resource. To reduce overhead, use XPARMS to disable calls for resources that you do not wish to protect. For example:

XTRAN

Transaction attach.

XPCT

Started transaction.

XPPT

Programs.

XFCT

Files.

XTST

Temp storage.

XDCT

Transient data.

If needed, you can utilize the application interface to protect a single resource. For example, if you only need to protect 1 program out of 5000, you can eliminate the overhead associated for checks related to the other 4999 programs by adding a TSSCAI call before going off to the new program.

Bypass and Protect Lists

The bypass and protect lists can be used in combination to selectively limit resource security in CICS. The bypass list provides the ability to bypass security resource checks. Resource names added to the bypass list are interpreted as generic prefixes.

The TRANID list, which is a subset of the BYPASS list, bypasses all security checks for an individual transaction. Entries specified in the TRANID bypass list bypass all security checks including OTRAN, LCF and FCT. This includes both LOCKTIME and job-submit processing.

The protect list forces a security check for a resource.

Note: TRANID does not bypass application interface calls.

Example: bypass and protect list

This example shows how the bypass and protect lists are used in conjunction to help tune your CICS security implementation:

```
TRANID bypass list: D
TRANDID protect list: DPAY
```

User enters:

```
DISP – resource check bypassed DABC - resource check bypassed
DPAY – resource check occurs
```

In this example, the TRANID bypass list is set to D. The TRANID protect list is set to DPAY.

A user attempting to enter transaction DISP or DABD will not incur a security check. But, if the user attempts to execute the DPAY transaction, a security resource check takes place.

CICS Cache

A CICS resource cache is an in-storage list of secured resources that a user has been allowed to access. The CICS cache reduces the number of host resource validation calls.

All successful resource checks for a user are cached. The cache exists for the life of the transaction. The CACHE box size is fixed at 512K. No auditing occurs for subsequent executions once the first transaction has been verified.

Two types of CICS cache are available:

Transaction life cache

Keeps a resource in the cache buffer for the life of the transaction. Once the transaction ends, the cache buffer is cleared. This is the default.

Session life cache

Keeps the resource in the cache buffer for the life of the signed on session. The cache is maintained until the user signs off. This provides improved performance since the authorization will be satisfied by the TSS CICS cache check. Set by control option OPTIONS(28).

Note: In addition to the time and date permissions, if auditing is in effect, only the first access for a particular resource is logged to the Audit Tracking file.

TSSCAI Application Interface

The Application interface RESLIST option provides performance improvement for client applications that request large amounts of CA Top Secret authorization data.

For example, if an application is building a menu that requires dozens of transaction calls to custom build a menu for each user, TSSCAI can extract in one call a complete list of authorized transactions. The application then scans the returned output from this single call. Both Permissions and Facilities can be extracted. Also, when performing a DUFXTR or DUFUPD, there is a cost of using TSSRNAME. If a TSSRNAME is specified, a sign-on takes place. If you do not specify a TSSRNAME, the request runs under the current signed on USER.

Chapter 10: Using the Sysplex Coupling Facility

This section contains the following topics:

The SYSPLEX XES Function

CA Top Secret takes advantage of the coupling facility functions:

XES Data Sharing

CA Top Secret uses this feature to provide the ability to share security file records for all CA Top Secret systems throughout the sysplex.The XES data sharing feature lets you define structures in the sysplex that multiple systems can use. These structures contain data (a security file for CA Top Secret) to be shared among the systems in the sysplex. Once the data has been placed in a structure it is valid to be read, updated, or deleted by any system connected to that structure. It is the responsibility of the systems programmer to define the structure-name to the coupling facility. This structure-name is then defined to CA Top Secret as the repository of the shared security file

XCF

CA Top Secret uses this feature as a message router that allows the communication of CA Top Secret security information between systems in the sysplex environment. It allows the propagation of commands issued on one system to all the other connected systems. XCF can be active when XES is not..

Note: A second structure can be defined for recovery purposes. This alternate structure is used in case the primary structure has faltered for any reason. This alternate structure is not mandatory, but can be used if desired and is also defined to CA Top Secret.

Structure Types

The coupling facility can define the structures:

- List
- Cache
- Lock

CA Top Secret and the SYSPLEX XES Function

The XES data-sharing feature lets you define a primary and an alternate list structure in the coupling facility.

If this feature is enabled, CA Top Secret attempts to obtain the requested data from the coupling facility before any physical I/O is performed. If the requested data is obtained from the coupling facility no I/O is performed. Thus saving the time from having to do the actual I/O to the database.

If you are using the CACHE facility with CA Top Secret, the cache call is performed prior to any call to the coupling facility. If the data is found in the CACHE, no calls to the coupling facility or the I/O to the database are performed.

Direct Physical I/O Requests

Whenever direct physical I/O is requested to the CA Top Secret database, a number of decisions must be made. It is important that any updating of any record is reflected on the database prior to the coupling facility. Most important is the need to obtain the data from the coupling facility when a direct READ of a record is requested. If any record is being updated, a lock to the coupling facility is issued to prevent the same record from being updated in the coupling facility by another CA Top Secret system. The security file ENQ record shows which system is currently holding the lock and a time stamp indicates the time the lock was placed.

List Structures

A list structure allows CA Top Secret to control what is placed in the coupling facility and the maintain the data once it is there. The primary list structure, defined by the systems programmer, is used to hold all the data for the CA Top Secret database. The optional alternate list structure is used if something happens to the primary list structure. If no alternate is defined, and the primary is unavailable, CA Top Secret defaults to its current I/O processing.

How List Processing Works

First define the list structure to the coupling facility. Then for each system enter a TSS MODIFY (SYSPLEX) command to define the connect-name, structure-names, and group-name, and to automatically connect each system to the coupling facility.

The security file ENQ record, the first physical record in the security file, keeps control information about the security file, including file locking status. The ENQ record includes the coupling facility structure name, and the coupling facility validity record includes the volume serial number (VOLSER) of the security file. This information is stored by the first system that connects to the coupling facility structure, and is used by systems that connect subsequently to ensure that a unique security file data set and structure name combination is used among all connected systems. When a local system attempts to lock the security file, the structure name in the security file ENQ record is compared to the local system's currently defined structure.

Whenever the security file is locked by a non-coupling facility connected system, the structure name found within the ENQ record is validated to be active, and the local system connects to the same structure, before proceeding with any update activity against the file.

While the current system is active in the coupling facility, if the security file ENQ record is altered by another system and it no longer contains the structure name, all systems are forced to disconnect from the coupling facility.

When a coupling facility connected system manually disconnects from the structure, all other connected systems are forced to disconnect as well, to ensure that no residual data is left in the coupling facility while one of the systems is running without an XES connection. An XES sysplex connection without an XCF connection is allowed.

After a system is forced to disconnect from a structure, as a result of a manual disconnect command, all other connected systems are forced to disconnect. No re-connect is attempted until a new structure is allocated in the coupling facility by a manual connect command, or by a new system starting up.

To determine if a structure was reallocated, use the TSS MODIFY(STATUS(SYSPLEX)) command, or the D XCF,STR z/OS operator command. Compare the current version number to the version number of the previous connection.

List Structures

CA Top Secret uses the list structure to “hold” the Security File. With a list structure, numerous functions can be performed against the security file, for example, reads, writes, deletes, and multiple deletes.

A connection between each system in the sysplex and the structure in the coupling facility must be created to establish communication. The connection is established prior to any other function and indicates to the coupling facility which structure contains the data for future processing.

When CA Top Secret is done communicating with the structure in the coupling facility a disconnect is performed. A list structure can be redefined while systems are connected to it. A redefining of the structure can take place if, for example, it becomes full and it is necessary to increase its size.

List Structure Components

A list structure is made up of headers and data elements. Each header can be used to break up different types of information in the structure. There is a maximum of sixteen headers that any one structure can have. With each header, data elements contain the data that is stored in the structure. The number of headers and the size of each data element are defined to the coupling facility when CA Top Secret “connects” to the structure. CA Top Secret only requires the use of one header. Multiple data elements can exist under that one header.

List Structure Size

The total size of the structure is dependent on how the structure is defined to the coupling facility. The required size of the coupling facility structure can be determined using the TSSFAR utility. This is done when the systems programmer defines the name of the structure. The amount of data is dependent on how the structure is defined at connection time. The first system to connect to a structure supplies information on how the structure is set up. For example, the number of headers and the maximum data element size, are two of many factors determining the amount of data that can be placed in a structure. These factors are supplied at connection time and are used in a calculation that is done during the connection.

List Structure Capacity

Once a list structure is full, no more data can be added to it. However, the system programmer can increase the size of the coupling facility structure with no disruption to the system or rebuild the structure characteristics with minimal disruption to the system.

Coupling Facility Error Handling

The coupling facility design handles any failure in it or its connections by disconnecting and reverting to normal I/O processing. After the problem is resolved, the user must reactivate the coupling facility by reactivating the sysplex connection.

The SYSPLEX XCF Function

The XCF function within a sysplex environment allows communication between all systems in the sysplex. The main function is the message routing capability. Any system within the sysplex can “join” a group and send and receive messages from each of the other members within the same group.

For example, if system A wanted to send a message to all other systems in the group it issues a send message to the coupling facility. The coupling facility is then responsible for posting or notifying all other “joined” systems within the group that a message is waiting for them. Each notified system is then responsible for receiving the message that was sent. The actions performed by the receiving system depend on the content of the message.

CA Top Secret and the SYSPLEX XCF Function

XCF message routing allows CA Top Secret operator commands to be sent to all other CA Top Secret systems in the sysplex that are defined to the same group. This allows one CA Top Secret system to update security control information and automatically send the information to the other CA Top Secret systems in the sysplex. This provides critical data synchronization on all systems without operator intervention.

The group-name, defined by the SYSPLEX control option, defines the group that a CA Top Secret system “joins” when the sysplex is started. If there are any other CA Top Secret systems that have joined the same group and the “send” command has been specified, the TSS command is sent to those systems. The TSS command is then processed on the remote CA Top Secret systems as if it was entered locally.

Note: In addition to being defined to the same group, the user must have CONSOLE authority on all other systems in the sysplex so that the TSS commands can be properly processed.

XCF(*) Control Option

XCF(*) is the CA Top Secret “send” command for routing information to remote systems in the sysplex. XCF(*) must the last parameter on the TSS MODIFY command.

Example: XCF(*)

This example updates the violation threshold on all systems in a group on a sysplex with the VTHRESH control option:

```
TSS MODIFY (VTHRESH(10,NOT),XCF(*))
```

If the command is successful, the following message is displayed at the sending system.

TSS9718I MODIFY COMMAND SENT VIA XCF TO ALL CONNECTED SYSTEMS IN THIS SYSPLEX

For information, see the *Control Options Guide.*

Controlling Access to XCF Policies

IBM provides an administrative utility, IXCMIAPU, to modify, add, or delete policy data from the ARM, CFRM, LOGR or SFM data sets. Use of this utility is controlled by the FACILITY class resource MVSADMIN. CA Top Secret implementation of the FACILITY class is done using the IBMFAC resource class.

Two access levels are available for these resources:

READ

Report only on a policy.

UPDATE

Alter and maintain a policy.

To secure access to the IXCMIAPU utility using CA Top Secret

1. Enter the command:

   ```
   TSS ADDTO(dept) IBMFAC(MVSADMIN)
   ```

   Ownership of the resource is established.

2. Enter the command:

   ```
   TSS PERMIT(user) IBMFAC(MVSADMIN.XCF.ARM)
                    ACCESS(READ | UPDATE)

   TSS PERMIT(user) IBMFAC(MVSADMIN.XCF.CFRM)
                    ACCESS(READ | UPDATE)

   TSS PERMIT(user) IBMFAC(MVSADMIN.XCF.LOGR)
                    ACCESS(READ | UPDATE)

   TSS PERMIT(user) IBMFAC(MVSADMIN.XCF.SFM)
                    ACCESS(READ | UPDATE)
   Permits to access the policy are set.
   ```

Sysplex to CA Top Secret Definition

To use the XCF or XES feature, the SYSPLEX control option must be entered. This control option contains the information necessary for CA Top Secret to communicate with the sysplex using the proper structure and group names.

For information, see the *Control Options Guide*.

Coupling Facility Management

z/OS and CA Top Secret use the definitions in the Coupling Facility Resource Management (CFRM) policy when managing the coupling facility storage.The CFRM policy resides in a CFRM couple data set which the system programmer creates using the IBM IXCL1DSU format utility program. To create the administrative CFRM policies, the systems programmer uses the IBM utility program IXCMIAPU. You can activate, change or rebuild the CFRM policy using the z/OS operator command SETXCF.

CFRM Policy

To set up the CFRM policy, the systems programmer must know which subsystems and applications in the installation will use the coupling facility and the structure requirements of each.

Use the IBM utility program IXCMIAPU to define the CFRM policy to:

- Identify each coupling facility and each structure.
- Specify the size of each structure. The structure size will initially be allocated at the INITSIZE you specify. The maximum size of the structure is defined by SIZE. Use the TSSFAR utility to run a TSSFAR ACIDCHAN report, which will recommend the XES structure size. You can use this to help determine the structure size you want to specify in the CFRM policy.
- Specify the preference list (PRELIST), which is an ordered list of coupling facility names in which the structure can be allocated.
- Specify an exclusion list of structures that are not to be allocated in the same coupling facility as the structure.

Every application specifies the required structure types, as well as the name and size of the structure or structures it uses. CA Top Secret uses the SYSPLEX control option to provide this information. For information on the SYSPLEX control option, see the *Control Options Guide*.

For information on the CFRM policy, see the IBM guide *MVS Setting Up a Sysplex.*

Coupling Facility Structure Size Alteration

CA Top Secret supports dynamic system-managed resizing or rebuilding of the coupling facility structure. You can use the SECXCF z/OS operator command to increase the coupling facility structure size with no disruption to the connected systems.

To increase the structure size up to the SIZE value, which is the maximum size defined in the active coupling facility resource management (CFRM) policy, enter:

```
SETXCF START,ALTER,STRNAME=strname,SIZE=nnn
```

If the optional initial size (INITSIZE) value was not specified in the CFRM policy, the structure size cannot be increased. In this case, the structure would have been initially allocated based on the maximum SIZE value, rather than the INITSIZE. If the structure is already at the maximum size specified in the CFRM policy, the CFRM must be modified to allow a larger structure size using the SETXCF START, REBUILD command.

When a structure ALTER processing completes, each connected system is posted by the CA Security Coupling Facility Service (CASECCFS) to allow those systems to update the size and utilization counts for the connected structure.

Display Coupling Facility XES Information

To display information about the current Coupling Facility XES structure, use the TSS MODIFY(STATUS(SYSPLEX)) command. The following sysplex related data appears in message TSS9661I:

CURRENT STRUCTURE SIZE

Current allocated size.

MAX STRUCTURE SIZE

Maximum structure size.

NUMBER OF STRUCTURE ENTRIES

Current entry count.

MAX NUMBER OF STRUCTURE ENTRIES

Maximum possible entry count.

Rebuild the Coupling Facility Structure

After you have updated the CFRM policy with new structure attributes such as SIZE or INITSIZE, you can use the SETXCF z/OS operator command to rebuild the coupling facility structure characteristics based on the new CFRM policy changes. The rebuild is performed with minimum disruption to the connected systems.

To initiate the rebuild process based on the new CFRM policy structure attributes, while connected systems remain connected to the structure, enter:

```
SETXCF START,REBUILD,STRNAME=strname
```

During the rebuild process, the system defers any requests that are issued while the structure is unavailable. These requests are processed after the rebuild process has completed.

Rebuild Requirements

The following requirements apply to the system-managed rebuild:

- Coupling facility has to be at CFLEVEL=8 or higher.
- The structure has at least two entries in the CFRM PREFLIST.
- All systems using the CFRM couple data set must be at OS/390 version 8 or higher.
- The CFRM couple data set must be formatted with the ITEM NAME(SMREBLD) NUMBER(1) statement.

Connecting to the Structure

At startup, or through a start command, CA Top Secret attempts to connect to the defined structures in the coupling facility. If successful, CA Top Secret attempts to use the coupling facility for all I/O direct requests being made for the defined file.

Define SYSTEM LOGGER to CA Top Secret

System logger is a z/OS component that allows an application to log data from different systems across a sysplex. A system logger application can be supplied by:

- IBM, for example the CICS log manager and the operations log stream (OPERLOG)
- Independent software vendors
- Your installation

A system logger application can merge the log data from systems across the sysplex into a log stream. A log stream is a collection of data in log blocks residing in a coupling facility list structure, on DASD, or on both.

To set up the z/OS CICS log manager with CA Top Secret

1. Follow substep A or B to define the IXGLOGE address space and the required SAF authorizations.
   a. Define IXGLOGR in the Started Task table to bypass SAF calls:

      ```
      TSS ADD(STC) PROCNAME(IXGLOGR) ACID(BYPASS)
      ```

      This bypasses all SAF calls and, therefore, does not require additional resource permits.

      or

   b. Define IXGLOGR in the Started Task table:

      ```
      TSS ADD(STC) PROCNAME(IXGLOGR) ACID(ixglogr)
      ```

2. Permit access to the log stream coupling facility structures.

   ```
   TSS ADD(dept) IBMFAC(IXLSTR.structure_name)
   TSS PER(ixglogr) IBMFAC(IXLSTR.structure_name) ACCESS(ALL)
   ```

3. Permit access to the DASD log stream and staging data sets.

   ```
   TSS ADD(dept) DSNAME(hlq.data_set_name)
   TSS PER(ixglogr) DSNAME(hlq.data_set_name) ACCESS(ALL)
   ```

   The default hlq is IXGLOGR if none is specified on the HLQ parm of the DEFINE LOGSTREAM statement.

4. Permit access to the systems SYSx.PARMLIB

   ```
   TSS ADD(dept) DSNAME(SYS1.PARMLIB)
   TSS PER(ixglogr) DSNAME(SYS1.PARMLIB) ACCESS(READ)
   ```

5. Control which applications can access the system logger resources.

   For logrec log stream, CICS log manager, and OPERLOG system logger application, define access to the log stream.

   ```
   TSS ADD(dept) LOGSTRM(log_stream_name)
   TSS PER(acid) LOGSTRM(log_stream_name) ACCESS(UPDATE)
   ```

6. Authorize who can use the IXCMIAPU utility.

   Define authorizations to use IXCMIAPU.

   ```
   TSS ADD(dept) IBMFAC(MVSADMIN.)
   TSS PER(acid) IBMFAC(MVSADMIN.) ACCESS(ALL)  <= Define structure
   TSS PER(acid) IBMFAC(MVSADMIN.) ACCESS(READ) <= Execute Report
   ```

   The LOGSTRM resource class is already pre-defined in CA Top Secret.

Appendix A: CERTADM Sample Code

The following is a copy of the CA Top Secret supplied command list, CERTADM. This list contains sample Digital Certificate commands that present the various Digital Certificate supported functional keywords.

```
/*===================================================================*/
/*Basic Self-signed Digital Certificate                              */
/*===================================================================*/
TSS CREATE(GENCDIV) TYPE(DIV) NAME('GENCERT DIVISION')
TSS CREATE(GENCDEPT) TYPE(DEPT) DIV(GENCDIV)
TSS CREATE(MARY001) NAME('GENCERT USER MARY') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS GENCERT(MARY001) DIGICERT(MARYCERT)
TSS LIST(MARY001) DATA(ALL,PASSWORD)
TSS REPLACE(MARY001) DIGICERT(MARYCERT) -
LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
TSS LIST(MARY001) LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
TSS LIST(SDT) DIGICERT(ALL)
TSS LIST(MARY001) SEGMENT(CERTDATA)
/*===================================================================*/
/*Create 5 Digital Certificates & add to the same user acid)         */
/*===================================================================*/
TSS CREATE(GENCDIV) TYPE(DIV) NAME('GENCERT DIVISION')
TSS CREATE(GENCDEPT) TYPE(DEPT) DIV(GENCDIV) -
NAME('GENCERT DEPARTMENT')
TSS CREATE(JAMES01) NAME('GENCERT USER JAMES') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS LIST(JAMES01) DATA(ALL,PASSWORD)
TSS GENCERT(JAMES01) DIGICERT(JIM01) LABLCERT('1ST D.CERT FOR JIM') -
KEYSIZE(512) KEYUSAGE(HANDSHAKE) ALTNAME('IP=203.9.102.100')
```

```
TSS LIST(JAMES01) DATA(ALL,PASSWORD)
TSS LIST(SDT) DIGICERT(ALL)

TSS GENCERT(JAMES01) DIGICERT(JIM02) LABLCERT('2ND D.CERT FOR JIM') -
NBDATE(10/01/02) NBTIME(08:00:00) -
NADATE(10/01/03) NATIME(09:00:00) -
KEYUSAGE(DATAENCRYPT) KEYSIZE(768) ALTNAME(DOMAIN=CA.COM) -
SUBJECTN('CN="JAMES SECOND DIGICERT"')

TSS LIST(JAMES01) DIGICERT(JIM02)
TSS LIST(SDT) DIGICERT(ALL)

TSS GENCERT(JAMES01) DIGICERT(JIM03) -
NBDATE(10/01/02) NBTIME(08:00:00) -
NADATE(10/31/03) NATIME(09:00:00) -
KEYSIZE(1024) -
LABLCERT('3RD D.CERT FOR JIM') -
KEYUSAGE(DOCSIGN) -
ALTNAME('IP=201.100.10.9 EMAIL=JAMES03@TEST.NET') -
SUBJECTN('T="THIRD BOOK OF JAMES" OU=PAYROLL')

TSS LIST(JAMES01) DIGICERT(JIM03)
TSS LIST(SDT) DIGICERT(ALL)

TSS GENCERT(JAMES01) DIGICERT(JIM04) -
SUBJECTN('CN="JIM DOUGLAS" O=CA ST="NEW JERSEY" C=US -
T="TEST GENCERT" L="NO. BRUNSWICK"') -
KEYSIZE(1024) -
LABLCERT('4TH D.CERT FOR JIM') -
KEYUSAGE(CERTSIGN) -
ALTNAME(URI=WWW.CA.COM)
TSS LIST(JAMES01) DIGICERT(JIM04)
TSS LIST(SDT) DIGICERT(ALL)
TSS GENCERT(JAMES01) DIGICERT(JIM05) -
SUBJECTN('CN=JIM05 O=CA ST=NJ C=US') -
NBDATE(10/01/02) NADATE(10/30/03) -
NBTIME(08:00:00) NATIME(09:00:00) -
KEYSIZE(4096) -
LABLCERT('5TH DIGICERT FOR JIM') -
ICSF -
KEYUSAGE(CERTSIGN) -
ALTNAME('IP=201.100.10.9 EMAIL=JAMES05@TEST.NET DOMAIN=CA.COM -
URI=WWW.CA.COM')

TSS LIST(JAMES01) DIGICERT(JIM05)
TSS LIST(JAMES01) DATA(ALL,PASSWORD)
TSS LIST(SDT) DIGICERT(ALL)
```

```
/*===================================================================*/
/*To Generate a Digital Certificate with keyword SIGNWITH & Remove     */
/* Digicert                                                          */
/*===================================================================*/
TSS CREATE(GENCDIV) TYPE(DIV) NAME('GENCERT DIVISION')
TSS CREATE(GENCDEPT) TYPE(DEPT) DIV(GENCDIV) -
NAME('GENCERT DEPARTMENT')
TSS CREATE(MARY001) NAME('GENCERT USER MARY') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS GENCERT(MARY001) DIGICERT(MARYCERT) –
    LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
TSS LIST(MARY001) DATA(ALL,PASSWORD)
TSS CREATE(TEDD001) NAME('GENCERT USER TEDD') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS LIST(TEDD001) DATA(ALL,PASSWORD)
TSS GENCERT(TEDD001) DIGICERT(TEDCERT1) -
SIGNWITH(MARY001,MARYCERT)
TSS LIST(TEDD001) DATA(ALL,PASSWORD)
TSS LIST(TEDD001) DIGICERT(TEDCERT1)
TSS LIST(SDT) DIGICERT(ALL)
TSS REMOVE(TEDD001) DIGICERT(TEDCERT1)
/*===================================================================*/
/*To Generate a Digital Certificate Request and write it to a data set.*/
/* (GENREQ)                                                          */
/*===================================================================*/
TSS CREATE(GENCDIV) TYPE(DIV) NAME('GENCERT DIVISION')
TSS CREATE(GENCDEPT) TYPE(DEPT) DIV(GENCDIV) -
NAME('GENCERT DEPARTMENT')
TSS CREATE(MARY001) NAME('GENCERT USER MARY') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS GENCERT(MARY001) DIGICERT(MARYCERT) –
    LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
TSS LIST(MARY001) DATA(ALL,PASSWORD)
TSS GENREQ(MARY001) DIGICERT(MARYCERT) -
DCDSN(QAPRN.GENREQ.MARYCERT) -
LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
TSS LIST(MARY001) LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
```

```
/*===================================================================*/
/*To Generate a Digital Certificate for a new acid, using the output   */
/* (DCDSN) from the GENREQ Statement above                             */
/*===================================================================*/
TSS CREATE(GENCDIV) TYPE(DIV) NAME('GENCERT DIVISION')
TSS CREATE(GENCDEPT) TYPE(DEPT) DIV(GENCDIV) -
NAME('GENCERT DEPARTMENT')
TSS CREATE(PAUL001) NAME('GENCERT USER PAUL') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS LIST(PAUL001) DATA(ALL,PASSWORD)
TSS GENCERT(PAUL001) DIGICERT(PAULCERT) -
DCDSN(QAPRN.GENREQ.MARYCERT) -
LABLCERT('LABEL FOR PAUL001 W/MARY"S DCDSN') -
SIGNWITH(MARY001,MARYCERT)
TSS LIST(PAUL001) LABLCERT('LABEL FOR PAUL001 W/MARY"S DCDSN')
TSS LIST(PAUL001) DIGICERT(PAULCERT)
TSS LIST(PAUL001) SEGMENT(CERTDATA)
/*===================================================================*/
/*To Generate a Digital Certificate for a user along with keyword SUBJECTN*/
/* And list the acid with different variations.                       */
/*===================================================================*/
TSS GENCERT(PAUL001) DIGICERT(PAULCT02) -
SUBJECTN('CN=PAUL O=CA OU="RESEARCH AND DEVELOPMENT"')
TSS LIST(PAUL001) -
SERIAL(00) ISSUERDN('.CN=PAUL.OU=RESEARCH AND DEVELOPMENT.O=CA')
TSS LIST(PAUL001) DIGICERT(PAULCT02)
TSS LIST(PAUL001) SEGMENT(CERTDATA)
TSS LIST(SDT) DIGICERT(ALL)

/*===================================================================*/
/*To EXPORT a Digital Certificate to an output data set NOT defined,   */
/* then do a CHKCERT command on the output DCDSN to verify that it     */
/* was EXPORTED.                                                      */
/*===================================================================*/
TSS LIST(MARY001) DIGICERT(MARYCERT)
TSS EXPORT(MARY001) DIGICERT(MARYCERT) -
DCDSN(QAPRN.OUTPUT.MARYCERT)
 TSS CHKCERT DCDSN(QAPRN.OUTPUT.MARYCERT)

TSS LIST(JAMES01) DIGICERT(JIM01)
TSS EXPORT(JAMES01) DIGICERT(JIM01) -
DCDSN(QAPRN.OUTPUT.JIM01) FORMAT(CERTDER)
TSS CHKCERT DCDSN(QAPRN.OUTPUT.JIM01)
TSS LIST(JAMES01) DIGICERT(JIM02)
TSS EXPORT(JAMES01) DIGICERT(JIM02) -
DCDSN(QAPRN.OUTPUT.JIM02) FORMAT(PKCS12B64) PKCSPASS(PSWDJIM2)
TSS CHKCERT DCDSN(QAPRN.OUTPUT.JIM02) PKCSPASS(PSWDJIM2)
```

```
TSS LIST(PAUL001) DIGICERT(PAULCT02)
TSS EXPORT(PAUL001) DIGICERT(PAULCT02) -
DCDSN(QAPRN.OUTPUT.PAULCT02) FORMAT(PKCS12DER) PKCSPASS(PSWDPAUL)
TSS CHKCERT DCDSN(QAPRN.OUTPUT.PAULCT02) PKCSPASS(PSWDPAUL)
/*===================================================================*/
/* Create Digital Certificate KEYRINGS and different variations of   */
/* the LIST command.                                                 */
/*===================================================================*/
TSS CREATE(GENCDEPT) TYPE(DEPT) DIV(GENCDIV) -
NAME('GENCERT DEPARTMENT')
TSS CREATE(MARY001) NAME('GENCERT USER MARY') TYPE(USER) -
PASSWORD(123,0)  DEPT(GENCDEPT)
TSS GENCERT(MARY001) DIGICERT(MARYCERT) –
       LABLCERT('SELF-SIGNED PRIVATE KEY FOR MARY')
TSS LIST(MARY001) DATA(ALL,PASSWORD)
TSS ADD(MARY001)  KEYRING(ACCOUNTG) LABLRING('ACCOUNTING-DEBT') -
RINGDATA(PAUL001, PAULCT02)  DEFAULT  USAGE(PERSONAL)
TSS ADD(MARY001)  KEYRING(ACCOUNTG) LABLRING('ACCOUNTING-DEBT') –
       RINGDATA(JAMES01, JIM02)   USAGE(CERTSITE)
TSS ADD(MARY001)  KEYRING(PERSONEL) LABLRING('PERSONEL-NEW HIRES') –
       RINGDATA(TEDD01, TEDCERT1)  USAGE(CERTAUTH)
TSS LIST(MARY001) KEYRING(ACCOUNTG)
TSS LIST(MARY001) SEGMENT(ALL)
TSS LIST(MARY001) DATA(ALL)
TSS LIST(MARY001) SEGMENT(CERTDATA)
TSS LIST(MARY001) SEGMENT(RINGDATA)

TSS LIST(SDT) KEYRING(ALL)
TSS LIST(SDT) DIGICERT(ALL)
TSS LIST(SDT) LABLRING('ACCOUNTING-DEBT')
```

Appendix B: RACF to CA Top Secret Translation

This section contains the following topics:

RACF to CA Top Secret Features

The following table compares RACF features with the corresponding CA Top Secret feature.

Feature	RACF	CA Top Secret
Users/Groups	SPECIAL Group concept	MSCA, LSCAs, Zones, Divisions, Departments, Profiles, ACIDs
Resource Protection	Data Sets	Data Sets (DSN)
	DASD volumes	DASD volumes (VOL)
	Tape volumes	Tape volumes (VOL)
	Terminals	Terminals (TERM, SOURCE)
	Load modules (programs)	Programs (PROG)
	IMS application group names (AIMS)	IMS applications (APPL)
	IMS transactions (TIMS and GIMS)	Transactions via Limited Command Facility (LCF) or protected resources (OTRAN)
	IMS applications	Facility checking
	CICS PSBs	CICS PSBs
	CICS transactions	Transactions via Limited Command Facility (LCF) or protected resources (OTRAN)

Feature	RACF	CA Top Secret
	CICS files	CICS files (FCT)
	CICS journals	CICS journals (JCT)
	CICS programs	CICS programs (PPT)
	CICS transient data destinations	CICS destinations (DCT)
	CICS temporary storage definitions	CICS temporary storage (TST)
	Installation-defined resources (CDT)	Installation-defined resources (RDT)
Access Authorizations	PERMIT command	TSS PERMIT command
	UACC	TSS PERMITs in the ALL Record
User Privileges	SPECIAL	MSCA and SCA with administrative authorities
	AUDITOR	SCA with MISC1, RES(REPORT) and DATA(ALL) authorities
	OPERATIONS	NODSNCHK, NOVLCHK, and NORESCHK attributes
	CLAUTH	Administrator with specifically named resource XAUTH authority
	GRPACC	PROFILEs
Administration	RACF commands	TSS commands
	ISPF panels	Panels for different facilities (TSO, CA-Roscoe, CICS and IMS)
Reports and Listings	RACF Report Writer	TSSUTIL, TSSTRACK
	DSMON Utility	TSSAUDIT, TSS LIST, TSS WHOHAS and TSS WHOOWNS
	SETROPS List Output	TSS LIST
	LISTUSER Output	TSS LIST
Customization	Exits	Exists and interfaces
Network Functionality	RRSF	Command Propagation Facility (CPF)

Feature	RACF	CA Top Secret
Other features	Userid identification	ACID identification
	Logging	Logging
	Started Task Class STARTED	STC facility

Commands

This section compares RACF commands with CA Top Secret commands.

ADDGRP

RACF uses the ADDGRP command is used to add a group definition. For example:

```
ADDGRP OMVSGRP OMVS(GID(1))
```

In CA Top Secret, this would be a profile record:

```
TSS ADD(OMVSGRP) GID(1)
```

ADDUSER

RACF uses the ADDUSER command to define a new user to its database and to define the profile information necessary to allow that user to use the desired components of the system. CA Top Secret does this using ACID and PROFILE records. The ADDUSER command is the same as the ADD command in CA Top Secret. For example:

```
ADDUSER USER01 DFLTGRP(OMVSGRP) OMVS(UID(200) HOME(/)
 PROGRAM(/bin/sh)) PASSWORD(password)
```

In CA Top Secret, this would be as follows:

```
TSS CREATE(USER01) TYPE(USER)
                    DEPARTMENT(dept1)
                    PASSWORD(password,0)

TSS ADDTO(USER01) GROUP(OMVSGRP)
                  UID(200)
                  HOME(/)
                  PROGRAM(/bin/sh)
```

ALTUSER

RACF uses the ALTUSER command to change an existing user's profile. For example:

```
ALTUSER USER01 CICS(OPCLASS(10) OPIDENT(U01) TIMEOUT(30))
```

In CA Top Secret, this is a change to the Acid record:

```
TSS ADD(USER01) OPCLASS(10)
                OPID(U01)
                OPTIME(30)
```

PERMIT

The RACF PERMIT command allows access to resources. The PERMIT command also exists in CA Top Secret. Use the TSS PERMIT command function to allow designated users to access the indicated data sets in an unlimited or a restricted manner. Restrictions are indicated by incorporating the appropriate PERMIT parameter.

This example allows USER01 to read data set SYS1.PARMLIB:

```
PERMIT SYS1.PARMLIB CLASS(DATASET) ID(user01) ACCESS(READ)
```

In CA Top Secret, this is a permit:

```
TSS PERMIT(user01) DSNAME(SYS1.PARMLIB)
                   ACCESS(READ)
```

RDEFINE

RDEFINE is used to define resources to RACF.

In CA Top Secret, the Resource Descriptor Table is a special ACID used to define resource classes and their properties. The RDT contains predefined TSS resource classes, including resources used by other CA product interfaces. It also contains dynamically defined resource classes. To administer the contents of the RDT Record, the TSS administrator can specify attributes, access levels, and default access levels using the TSS ADD command.

This example creates a user defined resource in CA Top Secret using the TSS ADD command:

```
TSS ADD(RDT) RESCLASS(xx)RESCODE(nn)
             ACLST(ALL=FFFF,UPDATE=6000,READ=4000,CREATE=1000,NONE=0000)
             ATTR(LONG)
```

CLASS

In RACF, with the exception of DATASET, USER, and GROUP classes, the entries in the class descriptor table (CDT) represent all resource classes for both z/OS and z/VM. The CDT consists of two modules: one is for IBM-supplied entries, the other is for installation-defined entries.

In CA Top Secret, all resources for both z/VM and z/OS are defined in the Resource Descriptor Table (RDT). Resources are predefined by CA Top Secret or dynamically defined by the particular installation.

RACF Attribute Translation

The following table shows how CA Top Secret translates a RACF attribute. CA Top Secret provides additional access levels for more granular control:

RACF Attribute	CA Top Secret Access Level
READ	READ
UPDATE	UPDATE
ALTER	ALL
CONTROL	CONTROL

Index

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z