CA Top Secret® for z/VM

Implementation Guide

r12

Fourth Edition

Contact CA Technologies

Contact CA Support

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

- 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.

If you would like 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 documentation updates have been made since the last release of this documentation:

- Password Phrase Defaults—Added minimum length.
- Appendix A—Added AT and VMRELOCA.

Contents

Chapter 3: System Entry Security 31

Chapter 4: Ownable Resource Security 47

Chapter 8: Implementing Ownable Resource Security 139

Chapter 9: Batch Operations 159

Chapter 1: ACID Security

This section contains the following topics:

Defining New User ACIDs

In general, User ACIDs are only used to designate specific employees within a department, and are therefore considered to be at the lowest level of the CA Top Secret organizational hierarchy. To define User ACIDs to CA Top Secret, a Security Administrator must specify the following keywords with the TSS CREATE command:

- NAME
- DEPT
- PASSWORD

TYPE(USER) does **not** have to be explicitly specified, since it is the system default. Below, is an example of how a new VM user might be defined:

```
TSS CREATE(USER01) NAME('ANDY PARKS') DEPT(COMPLAINT)
   PAS(INITIAL,7,EXPIRED) FAC(VM)
```

Note: In order for an administrator to define this user, he **must** have ACID(CREATE) authority and a scope that encompasses the specified department.

Defining Profile ACIDs

Often, a group of users will need to use a set of identical resources in the same way. Since administrators should try to minimize the use of access authorizations made directly to users, it is a good idea, instead, that they define common resource access characteristics, or profiles, which can then be assigned to as many users as is appropriate. By using profiles, administrators can do away with redundant authorizations, thereby making it easier to administer the security database.

A maximum of 254 profiles may be attached to a single user. Profiles cannot be assigned passwords, nor can they be attached to other profiles.

In order to define a profile, the following keywords must be specified with the TSS CREATE command:

- TYPE
- NAME
- DEPT

Note: The DEPT keyword is only used if the Control ACID issuing the command is an SCA, LSCA, ZCA, or VCA.

For example, if an administrator wants to create a Payroll Department profile, he would enter:

```
TSS CRE(PAYPR01) TYPE(PROF) NAME('PAYROLL DEPT PROF') DEPT(PAYROLL)
```

He could then use the PERMIT function to permit resource access authorizations to the profile:

```
TSS PERMIT(PAYPR01) VMMD(USER01) ACC(READ)  TIME(08,19)
```

Attaching a Profile to an ACID

Once a profile has been defined, it can then be attached to an ACID. To do this, either the TSS CREATE or TSS ADDTO command can be used. For example, the following adds the authorizations granted to PAYPR01 to USER01:

```
TSS ADD(USER01) PROFILE(PAYPR01)
```

Note: An administrator **must** possess ACID(MAINTAIN) authority to attach a profile to any ACID that lies within his scope. The profile itself must also fall within the administrator's scope, unless it is a globally administrative profile, in which case it can be attached by any administrator, to any user within his scope.

Ordering of Profiles Attached to an ACID

Because the order of profiles directly affects the permissions granted, administrators should make sure that all profiles are ordered as they want them to be. By default, any new profiles added to a list of existing profiles, will be added to the end of the list. With the TSS ADDTO command, however, administrators can designate where in the list, a new profile is to be added. To alter the desired order of the new profile, the following syntax should be used:

```
TSS ADD(acid) PROFILE(profacid) AFTER|BEFORE(existingprofacid)
```

For example, suppose an ACID called USER01 currently has four profiles: PROF1, PROF2, PROF3, and PROF4. If an administrator wants to add a new profile, PROF5, before PROF3, he could enter either of the following:

```
TSS ADD(USER01) PROFILE(PROF5) AFTER(PROF2)
            or
TSS ADD(USER01) PROFILE(PROF5) BEFORE(PROF3)
```

Note: Already existing profiles cannot be reordered, only newly added ones can.

You can also add an existing profile and make it the first one in the search order by using the FIRST keyword. For example, to add an existing profile called PROF4 to USER04 and make it the first in the search order, issue the command below.

```
TSS ADD(USER04) PROFILE(PROF4) FIRST
```

Globally Administrative Profiles

A conventional departmental profile can only be attached to users if both the user and the profile fall within the administrator's scope. In some cases, however, it is desirable to create a profile that might be available to all administrators. Globally Administrative Profiles, or GAPs, provide this capability. Giving the GAP attribute to a profile allows any security administrator to attach the profile to any user within his administrative scope.

For example, if the DCA DPT1ADM creates the profile PROF01, then he can permit global administration by entering:

```
TSS ADD(PROF01) GAP
```

Now any administrator can attach PROF01 to any user within his scope.

To remove the GAP attribute from a profile, the REMOVE command function can be used. For example:

```
TSS REM(PROF01) GAP
```

Assigning Profiles Outside Your Scope

To assign an ACID scope over a profile, use the ADMIN command function. The following example assumes that DCA2 administers DEPT2 and that PROF1 resides in DEPT1. To give DCA2 administrative authority over PROF1, DCA1 can issue this command:

```
TSS ADMIN(DCA2) SCOPE(PROF1)
```

Note: Administration of profiles may not be given to ACIDs with the type of USER.

Profile Expiration

If an administrator wants to temporarily add a profile to a User ACID, he may do so by using the FOR or UNTIL keywords. For example, if an administrator enters the following, then the User ACID will only be connected to the specified profile for the next ten days:

```
TSS ADD(acid) PROFILE(prof) FOR(10)
```

At that time, CA Top Secret will automatically remove the profile from the User ACID.

To list the expiration interval for all temporary profiles that have been added to a User ACID, an administrator can specify the EXPIRE operand, which can be used with the DATA keyword of the TSS LIST command.

Defining Department, Division, and Zone ACIDs

Other ACID types that may be defined through CA Top Secret include Department, Division, and Zone ACIDs. Department ACIDs are used to define resources to entire departments, while Division ACIDs are used to define resources to entire divisions. Zone ACIDs, on the other hand, are used to define resources to two or more divisions that have been grouped together.

Only two keywords are required to define Department, Division, or Zone ACIDs:

- TYPE
- NAME

For example, to create the Financial Applications department, an administrator would enter:

```
TSS CRE(FINDEPT) TYPE(DEPT) NAME('FINANCIAL APPL')
```

To associate the newly created Financial Applications department with the Financial Applications division, an administrator would enter:

```
TSS CRE(FINDEPT) TYPE(DEPT) NAME('FINANCIAL APPL') DIV(FINDIV)
```

Finally, to associate the Financial Applications division with the Financial Applications zone, he would enter:

```
TSS CRE(FINDIV) TYPE(DIV) NAME('FINANCIAL DIVISION') ZONE(FINZONE)
```

Note: A Department ACID cannot be directly attached to a Zone ACID. It must first be attached to a Division ACID which can then be attached to a Zone ACID.

Defining New Control ACIDs

Control ACIDs, because they are used to define Security Administrators, are considered to be at the top of the CA Top Secret organizational hierarchy. To define a new Control ACID, the following keywords must be specified:

- NAME
- PASSWORD
- TYPE (i.e. SCA, LSCA, ZCA, VCA, DCA)
- *scope* (i.e. the appropriate department for a DCA)

For example, the following defines a VCA, called DEVVCA, to the Development division:

```
TSS CRE(DEVVCA) TYPE(VCA) NAME('DEVELOPMENT VCA')
   DIV(DEV) PASS(PEAR01,7)
```

If a new DCA was being defined, then DEPT would replace the DIV keyword. If a new ZCA was being defined, then ZONE would replace the DIV keyword. If a new SCA was being defined, then neither the DIV nor DEPT keywords would be necessary, since, by default, an SCA's scope encompasses the entire corporation.

Defining Model ACIDs

Security Administrators can use templates or model ACIDs to create new ACIDs. All basic information, including the name associated with the ACID, the ACID type, permitted facilities, and associated profiles can be copied from the model ACID to the new ACID. Administrators have the ability to change, add, or omit any of this information. In addition, all ACID types can be created using this feature. For example, a model ACID can be created using the following command:

```
TSS CRE(MODELUSR) NAME('Model User') TYPE(USER)
   PAS(password) INSTDATA('test data') DEPT(deptacid)
```

It is not necessary to create an ACID to be used as a template, since any existing ACID can be used. To create an exact copy of the model ACID, an administrator would enter:

```
TSS CRE(newacid) USING(MODELUSR)
```

To create an exact copy but with new installation data, an administrator would enter:

```
TSS CRE(newacid) USING(MODELUSR) INSTDATA('new information')
```

This flexibility in creating new ACIDs applies to all basic information fields, with the exception of attributes. Attributes can be added, through the TSS CREATE USING command, but cannot be omitted.

ACID Creation Requirement Chart

The following table shows the requirements for creating the eleven types of ACIDs. The table also provides some additional options, which are in square brackets.

ACID Type	Model TSS CREATE Entry
SCA	`TYPE(SCA) NAME('name') PAS(password) [FACILITY(facilities)]`
LSCA	`TYPE(LSCA) NAME('name') PAS(password) [FACILITY(facilities)]`
ZONE	`TYPE(ZONE) NAME('name')`
ZCA	`TYPE(ZCA) NAME('zone') PAS(password) ZONE(zone) [FACILITY(facilities)]`
DIVISION	`TYPE(DIVISION) NAME('name')[ZONE(zone)]`
VCA	`TYPE(VCA) NAME('name') DIVISION(division) PAS(password) [FACILITY(facilities)]`
DEPARTMENT	`TYPE(DEPT) NAME('name') [DIVISION(division)]`
DCA	`TYPE(DCA) NAME('name') DEPT(department) PAS(password) [FACILITY(facilities)]`
PROFILE	`TYPE(PROFILE) NAME('name') DEPT(department)`
GROUP	`TYPE(GROUP) NAME('name') DEPT(department)`
USER	`NAME('name') DEPT(department) PAS(password) [FACILITY(facilities)]`

Deleting ACIDs

The DELETE function purges the specified ACID from the CA Top Secret Security File. For example, the following deletes the ACID USER999:

```
TSS DELETE(USER999)
```

All authorizations granted to USER999 are automatically revoked and all resources on USER999 are automatically freed.

Note the following restrictions regarding the use of this function:

- To issue this function, the administrator must have ACID(CREATE) authority and an administrative scope that includes the ACID.
- If the ACID still owns resources that have been PERMITted to other users, it cannot be deleted.
- If the ACID is still PERMITted to other ACIDs, it cannot be deleted.

- A zonal, divisional, departmental, or profile ACID that still has other ACIDs linked to it cannot be deleted.
- The ALL and AUDIT Records, as well as the MSCA's ACID, can never be deleted.

Setting ACID Expiration Dates

When used with either the CREATE or ADDTO functions, the FOR or UNTIL keywords limit how long a User ACID remains valid. The FOR keyword specifies how many days an ACID is valid, starting on the day the FOR keyword is first entered. For example, the following limits USER01's access to the system for the remainder of the day:

```
TSS ADD(USER01) FOR(1)
```

FOR(2) grants access rights for the remainder of the day, plus all of the next day, and so on.

The UNTIL keyword instructs CA Top Secret to automatically expire the designated ACID on the indicated date. For example, the following allows for the use of USER01 up to but not including December 31, 2002:

```
TSS ADD(USER01) UNTIL(12/31/02)
```

Note that these two keywords can also be used with the PERMIT function to specify a time limit on authorized access to the designated resource(s).

Reactivating an Expired ACID

To reactivate an expired ACID, the FOR keyword can be incorporated into a REMOVE function with a "0" operand. For example, the following reactivates USER01.

```
TSS REM(USER01) FOR(0)
```

The same method can also be used to reactivate an ACID that expired as the result of an UNTIL keyword.

Changing an ACID's Expiration Date

The REPLACE keyword can be used to change the duration specified by a FOR or UNTIL keyword. For example:

```
TSS REP(USER01) UNTIL(12/25/03)
```

would allow for the use of USER01 up to December 25, 2003, instead of December 31, 2002 as specified above.

Setting Inactivity Suspensions for ACIDs

The INACTIVE control option can be set to suspend any ACID that has not been used for a long period of time. Security Administrators may specify that, after 0 to 255 days, an inactive ACID automatically be suspended. Inactivity is measured from the day that an ACID's password expires. If CA Top Secret does not detect any activity for this ACID before the inactivity threshold is reached, the ACID is suspended. Note that changing the password is considered activity.

Note: A Security Administrator with ACID(MAINTAIN) authority and the appropriate scope can reactivate a suspended ACID by removing the SUSPEND attribute from the user.

Setting Limited-Duration Suspensions for ACIDs

With the SUSPEND keyword, Security Administrators can instruct CA Top Secret to suspend an ACID until a specified date--at which time the ACID will automatically be reactivated. For example, the following suspends USER01 until July 5, 2002, when it will automatically be reinstated:

```
TSS ADD(USER01) SUSPEND UNTIL(07/05/02)
```

If USER01 happens to be logged on at the time, then the suspension begins when they log off. (To suspend a user immediately--even if logged on--use TSS MODIFY SUSPEND(acid)). This feature simplifies the process of temporarily deactivating ACIDs during vacations, extended trips, and leaves of absence, etc.

Displaying Information About ACIDs

TSS LIST can be used to display information about an ACID. The source of this information is the ACID's Security Record. Since this information might be sensitive, CA Top Secret strictly controls who can see what information. Exactly what information can be LISTed depends on the ACID's administrative authorities and scope of authority.

To obtain any information from the Security Record, the ACID issuing the LIST must possess DATA authority. What information users can see depends on which DATA authorities they have, such as DATA(BASIC) or DATA(RESOURCE). For a complete list of the various DATA authorities, refer to the *Command Functions Guide*.

For example, one of the DATA options is NAMES. This option can be used to list the names of all users in the department. For example, a DCA with DATA(NAMES) authority could enter the following TSS command to view the list of names:

```
TSS LIST(ACIDS) DATA(NAMES)
```

To obtain basic information about every user within the department, a Security Administrator must have DATA(BASIC) authority. They could then enter the following, which returns information about the names, types, facility, and/or restrictions on the users who are within the issuing administrator's scope of authority.

```
TSS LIST(ACIDS) DATA(BASIC)
```

For instance, a DCA can receive information about the people in his department, while a VCA can receive information about the people in his division.

To obtain installation-wide information, the MSCA or an authorized SCA must issue the LIST.

Allowing Users to LIST Their Security Records

To give users the authority to display their Security Record information (except for password information and connected profiles), the MSCA or an SCA with DATA(ALL) authority must enter the following TSS command:

```
TSS ADMIN(ALL) DATA(ALL)
```

USER01 may then list all of his Security Record, except password and profile information, by entering:

```
TSS LIST(USER01) DATA(ALL)
```

Chapter 2: Password and Passphrase Security

This section contains the following topics:

Defining Passwords

CA Top Secret utilizes password security as a means of protecting User and Control ACIDs from unauthorized access. A wide variety of password security policies can be implemented through the password protection controls and options provided by CA Top Secret. Although password validation can be overriden for selected ACIDs, the use of password security is highly recommended and encouraged.

The main vehicle for defining passwords to CA Top Secret is the PASSWORD keyword. The PASSWORD keyword is used in conjunction with TSS commands. It allows administrators to specify a password, an optional automatic expiration interval or a statement that can expire the password the first time it is used. For example, the following assigns the User ACID, USER01, the password HAPPY, which will expire the first time it is used.

```
TSS CRE(USER01) NAME('MAX SMILEY') DEPT(FINANCE)
   FAC(VM) PASSWORD(HAPPY,,EXP)
```

Since no expiration interval has been specified, the succeeding password will expire in the system default value, set by the installation.

Note: The PWEXP control option can be used to specify a default password expiration interval for new users. This interval can range from 0 - 255. Specifying 0 indicates that passwords for new users will never expire.

If an administrator chooses to specify an expiration interval for USER01, then they should enter the following which would set the password expiration interval at the maximum of 255 days.

```
TSS CRE(USER01) NAME('MAX SMILEY') DEPT(FINANCE)
   FAC(VM) PASSWORD(HAPPY,255)
```

If no password is required for USER01, then the NOPW option should be used instead:

```
TSS CREATE(USER01) NAME('MAX SMILEY') DEPT(FINANCE)
   FAC(VM) PASSWORD(NOPW)
```

Unless otherwise specified, CA Top Secret will automatically assume that a user's password applies to all facilities to which that user has access. However, if that is not to be the case, then the MULTIPW keyword can be used. The MULTIPW keyword can be set to allow different passwords to be used for different facilities. For example, the following example assigns the indicated passwords to the respective facilities.

```
TSS CRE(USER01) NAME('MAX SMILEY') DEPT(FINANCE)
   FAC(VM) PAS(HAPPY) MULTIPW

TSS ADD(USER01) PAS(SUNNY) FAC(VMTEST,VMPROD)
```

Note: Remember, an administrator **must** have ACID(CREATE) authority and an administrative scope that includes the specified department, in order to define any user.

Password Controls

CA Top Secret uses a number of password controls to determine who can specify passwords, how they are to be generated, what form they must take, and how long they are to remain valid. These controls are determined through the CA Top Secret control option settings.

Changing Passwords

By default, CA Top Secret allows users to change their passwords. Whenever a password is changed, CA Top Secret updates the user's Security Record. Passwords may also be changed by security administrators, through the TSS ADDTO and REPLACE functions. For example, suppose a security administrator wants to change USER01's password from SUNNY to CLOUDY. He would enter:

```
TSS ADD(USER01) PAS(CLOUDY)
```

Note: Remember, an administrator must have ACID(MAINTAIN) authority and the necessary administrative scope, in order to change a user's password.

Maintaining Password History

Through the PWHIST control option, a security administrator can specify the number of previous passwords that are to be maintained as part of an ACID's password history. The reason for maintaining a password history is to prevent users from reusing old passwords everytime their current password expires. The number of old passwords that can be maintained by CA Top Secret ranges from 0 to 64. Specifying 0 indicates that no password history is to be maintained and that the user is able to make his new password identical to his current password.

CA Top Secret will reject all identical passwords going back as far as the specified PWHIST value. The value specified for PWHIST will, however, have no affect on ACIDs with the MULTIPW attribute, since those ACIDs have an internal PWHIST value of 3.

Specifying Passwords

Security administrators can select from an extensive group of password control options to establish installation wide password change rules. CA Top Secret defaults to the following rules for password changes:

- The new password cannot be the same as the current password
- The new password cannot be exactly the same as any of the user's previous passwords (up to 64)

Additionally, if the NEWPW control option is left at its default, the following rules will be in effect:

- The new password cannot be a close variant of the user's previous password: NEWPW(TS) It is considered too similar if:
  - The first three characters are identical
  - The second three characters are identical
  - The last three characters are identical
  -
- Passwords can not be changed more than once per day: NEWPW(MINDAYS=1) This interval does not affect:
  - SCAs and LSCAs who can change passwords as often as they want
  - Users who select the random password feature and can change their passwords as often as they want
- Passwords must be at least four characters: NEWPW(MIN=4)
- Passwords cannot contain repeating characters: NEWPW(NR=0)

- Passwords that match entries in a restricted password list are not allowed
- Passwords that match the userid or the first four characters of any word in the associated personal name field are not allowed: NEWPW(ID)

These, and additional rules can be implemented through the NEWPW control option. Refer to the *Control Options Guide* for a complete list.

Password Expiration Intervals

A password expiration interval is the number of days before CA Top Secret forces a user to change his password. This interval can range from 0 to 255 days. If 0 is specified, then the user never has to change his password.

When a user's password is about to expire, CA Top Secret will issue a PASSWORD EXPIRATION WARNING message on the job log or terminal. Once the user's password expires, the user must provide a new password before he can sign on to CA Top Secret again. A user may change his password before, on, or after the day it expires.

Administrators can use the password expiration interval to:

- Set the password expiration interval on a user-to-user basis, from 0 - 255 days
- Specify EXP so that a user's password will expire immediately upon logon
- Specify the minimum number of days that a user must wait after changing his password, before he may change it again (MINDAYS=nn).

These specifications help to minimize the chances of password exposure or disclosure.

Prohibited Passwords and Prefixes

CA Top Secret provides a list of prohibited prefixes for passwords. This list prevents the use of obvious passwords, such as the user's or company's name, the name of the month, or the name of facilities. This list is only in effect for new passwords which are entered while the NEWPW(RS) control option is in effect. Administrators can add any number of prefixes to this list, either temporarily or permanently.

To temporarily add prefixes to the list, an administrator would use the RPW control option. For example, if an administrator wants to add the prefix XYZ to the restricted prefix list, he would enter:

```
TSS MODIFY(RPW(ADD,XYZ))
```

If he wants to remove the prefix XYZ from the list, then he would enter:

```
TSS MODIFY(RPW(REMOVE,XYZ))
```

To view all prefixes currently included in the list, an administrator would enter:

```
TSS MODIFY(RPW(LIST))
```

To make permanent changes to the restricted password list, administrators need to modify the Parameter File.

Password Violation Threshold

CA Top Secret uses a password violation threshold to prevent users from continually guessing at an ACID's password. This threshold is set with the PTHRESH control option. Once the PTHRESH option has been set, CA Top Secret will count each incorrect password attempt. The password violation threshold will be applied system-wide, and can range from 1 - 254. The default is 3.

Once the user enters the correct password for the ACID, CA Top Secret will reset the violations counter to 0. If the user exceeds the password violation threshold, then the ACID will be suspended. A suspended ACID can only be reinstated by a security administrator, using the TSS REMOVE SUSPEND function. Again, once the ACID is no longer suspended, CA Top Secret will reset the violations counter to 0. For more information on the PTHRESH control option, refer to the *Control Options Guide*.

Password Masking

Password masking allows CA Top Secret to control the type of passwords that users can create. Password masks are set through the NEWPW control option, and will apply to all new passwords, including those that are randomly generated. Password masks are enforced system-wide for all users except the MSCA, who is exempt.

The character types used to define the mask include:

Symbol	Meaning
C	Consonant
V	Vowel
N	Numeric character
X	Non-vowel
?	Any character

For example, the following would allow (or randomly generate) passwords such as TIGER or MONET.

```
NEWPW(MASK=CVCVC)
```

If all eight-character positions are not defined, then users are unrestricted. For instance, in the example above, the CVCVC only accounts for five of the eight character positions. Therefore, users would only have to follow the mask for the first five characters, but could specify whatever they wanted for the last three. Thus, passwords such as TIGER123 or MONETABC would also be valid.

Random Password Generation

By using the RANDOM option, when prompted for a password, a user can invoke CA Top Secret to generate a random password.

Note: This option only works when the RNDPW facility control option is ON.

Random passwords can be generated by:

- A user requesting a random password, or
- A security administrator instructing CA Top Secret to generate a random password for any user whose password has expired

All random passwords will conform to the designated password masks and will also be checked against the restricted password list.

Requesting a Random Password

To automatically generate a random password, users can specify the RANDOM option. The randomly generated password will appear on the screen until the screen is cleared. The NEWPW(MINDAYS=nn) restriction does not apply to randomly generated passwords.

Automatic Password Generation After Expiration

CA Top Secret can be requested to randomly generate a new password whenever a user attempts to signon after his password has expired. This capability is turned on through the NEWPW control option, by specifying NEWPW(RN).

Note: This option only works when the RNDPW control option is turned ON.

Users whose passwords have been automatically generated by CA Top Secret, can still change passwords at any time, unless prevented from doing so by NOPWCHG or NEWPW(NU), discussed on the following page.

Preventing Users From Changing Passwords

Although, by default, CA Top Secret allows users to change their passwords, this capability may be revoked system-wide, or revoked for certain users or user groups. To prevent all users in the system from changing their passwords, administrators should set the NU suboption of the NEWPW control option as follows:

```
NEWPW(NU)
```

To prevent specific users from changing their passwords, administrators should use the NOPWCHG keyword with either the CREATE or ADDTO functions. This keyword can be attached to either User ACIDs or profiles. For example, the following would prevent USER01 from changing his password:

```
TSS ADDTO(USER01) NOPWCHG
```

Forcing Password Validation in DORMANT and WARN Modes

In DORMANT mode, CA Top Secret does not perform security validation for normal users (CA Top Secret always performs password validation for Security Control ACIDs). In WARN mode, CA Top Secret performs security validations for all users, but does not deny them access to the resource they are requesting. Instead, it sends a simple violation message. However, with the WARNPW suboption of the FACILITY control option, password validation can be forced in these modes. For example, the following turns this option on, in WARN mode, for VMTEST:

```
FACILITY(VMTEST=WARNPW)
```

A similar option, DORMPW, forces users to enter their CA Top Secret password in DORMANT mode.

Password Phrase Controls

CA Top Secret uses a number of password phrase controls to determine who can specify password phrases, how they are to be generated, what form they must take, and how long they are to remain valid. These controls are determined through the control option settings.

Passwords and Password Phrases

CA Top Secret for z/VM requires password protection for all ACIDs by default. In addition to a password, an ACID can have an optional password phrase. You can use password phrases instead of passwords in applications that support them.

Passwords have a maximum length of eight characters. A password phrase can be from 14 to 100 characters long and can include mixed-case letters, numbers, and special characters including blanks. The same user ID can have a password for applications that accept passwords only and a password phrase for other applications.

The security administrator can specify:

Password and password phrase settings, including:

- Minimum length
- Permitted and required content
- Expiration interval
- whether passwords are for all facilities or individual facilities.
- whether CA Top Secret for z/VM prompts users for new password and password phrase verification.

Password Phrase Defaults

Set password phrase defaults with the PPHRASE, NPPTHRESH, PPEXP, PPHIST, and NEWPHRASE control options. The defaults are:

The password phrase:

- Must be at least 14 characters long
- Must be at least 9 characters long
- Can be up to 100 characters long
- Expires after 30 days
- Cannot be the same as the previous three password phrases

CA Top Secret for z/VM issues warning messages three days before the password phrase expires.

Users cannot change a phrase more often than once each day (except for security administrators).

To manage this feature, see the *Control Options Guide*.

Chapter 3: System Entry Security

This section contains the following topics:

Controlling Access to VM

As supplied, virtual machines running on a CPU with CA Top Secret are doing so under the default facility--VM. CA Top Secret controls access to VM in FAIL mode by requiring that the user be authorized to use the facility. By default, only the MSCA can access VM when CA Top Secret is first installed with an empty security file. Everyone else must be explicitly authorized to use the VM facility through a TSS CREATE or ADDTO function.

To authorize users for access to the VM facility, CA Top Secret administrators (other than the MSCA) need FACILITY(VM) administrative authority and the appropriate scope. If the Security Administrator has FAC(ALL) administrative authority together with a scope encompassing USER01, then this TSS command may be issued:

```
TSS ADD(USER01) FAC(ALL)
```

To define other facilities, you can edit the Parameter File. For instance, suppose you wanted to distinguish your Test and Production VM systems. Begin by defining the VM facility. To do so, use the FACILITY control option in the Parameter File:

```
FACILITY(USER1=NAME=VMTEST)
FACILITY(VMTEST=MODE=WARN)
```

where:

- FACILITY—control option
- USER1—user-definable Facilities Matrix entry
- NAME—renaming keyword for FACILITY control option
- VMTEST—name of the new facility

- MODE—keyword for setting security mode
- WARN—user-defined security mode affecting VMTEST facility

Next, use the VMFAC control option to associate your CA Top Secret facility with a counterpart SYSID:

```
VMFAC(SYSTEMC=VMTEST)
```

where:

- VMFAC—control option
- SYSTEMC—systemid from system_identifier statement in the system configuration file
- VMTEST—the test machine you set up earlier.

For more information about these terms, see the *Control Options Guide*.

As you can see, CA Top Secret is flexible enough for a production VM environment. Just as you can set up a test machine, you can also set up other test and production machines.

To prevent suspected subversion, CA Top Secret allows the VM facility to be immediately **deactivated** from the VM operator's console. While the facility is inactive, no logons are accepted. Security Administrators can deactivate the VM facility by using a TSS MODIFY command to specify the INACT (inactive) suboption of the FACILITY control option. The facility may be reactivated with the ACTIVE suboption of FACILITY. Here's an example of inactivating and reactivating the VM facility:

```
TSS MODIFY FAC(VM=INACT)
```

-and-

```
TSS MODIFY FAC(VM=ACTIVE)
```

Controlling Access to CPUs

Security Administrators may use CA Top Secret to protect a specific CPU (identified in the system configuration file) once it is owned. Use the CPU keyword, along with TSS CREATE or ADDTO, to establish resource ownership.

Once owned, a CPU cannot be accessed unless explicit authorization is granted. The CPU keyword is the vehicle for specifying CPU ownership and authorizations.

For example, the following assigns ownership of this CPU to the department associated with ACID DEPT01:

```
TSS ADDTO(DEPT01) CPU(VMSYSA)
```

Remember that CPU is a “resource class” keyword--just like CPCMD, VOLUME, or VMMDISK.

To grant USER01 access to VMSYSA, a Security Administrator with the required administrative authority enters:

```
TSS PERMIT(USER01) CPU(VMSYSA)
```

By qualifying a TSS PERMIT with other attributes, CPU protection can be used to limit access to a CPU on a time-of-day basis. For example, to provide a virtual machine that can be accessed by the day shift, a Security Administrator enters a TSS command similar to this example:

```
TSS PERMIT(DAYUSER) CPU(VMSYSA) FACILITY(VM) TIME (08,17)
```

Suppose you restrict access between 8:00:00 and 17:00:00 hours, as in the last example. To compensate for a three-hour time difference, use TZONE:

```
TSS ADDTO(USER01) TZONE(-3)
```

This example shows how you compensate for a time difference of three hours. In this case, you know that the user is to your west because of the “minus” sign. Remember, minus sign means west; plus sign means east. TZONE may be added or created at the user or profile level.

Controlling Access to Terminals

Terminals can be protected in three ways:

- By forcing a user to log on from a specific terminal
- By restricting use of terminals in sensitive areas

- By controlling who can DIAL a particular line on another virtual machine

Online terminals can be defined to CA Top Secret and then have their access selectively authorized. The TERMINAL keyword is used to establish access authorizations and restrictions. To identify an online VM terminal the following conventions are used:

TYPE	PREFIX	EXAMPLE
locally attached	GRAF plus four character local address	`TSS ADD(BUDDEPT) TERM(GRAF02BA)`
remotely attached VM-controlled network terminals	NETW plus four character resource id	`TSS ADD(CORP) TERM(NETW0301)`
logical devices	LDEV plus four character address of logical device which is arbitrarily defined	`TSS ADD(CORPNET) TERM(LDEV1234)`
VTAM/SNA	8 character LU name	`TSS ADD(FINDEPT) TERM(`*xxxxxxxx*`)`

Terminal access protection can be combined with facility limitations, time restrictions, DIALed terminals, etc. to further tailor system entry control.

The CA Top Secret resource class keyword TERMINAL is used to refer to both local and VTAM terminals.

Using the TERMINAL Keyword

The TERMINAL parameter is used to define terminals to the system and to establish access authorizations. If a generic prefix is used, its length must be between one and eight characters. For example, to include ownership of VTAM terminal L1032209 in the accounting department's scope, enter this TSS command:

```
TSS ADD(ACCTDEPT) TERM(L103)
```

Once defined to CA Top Secret, terminals (who can access them) are specifically restricted. For example, the following authorizes USER01 to use any L103 terminals in the morning:

```
TSS PER(USER01) TERM(L103) TIMES(07,12)
```

To give all users this kind of authority, enter:

```
TSS PER(ALL) TERM(L103) TIMES(07,12)
```

To allow a specific user to access all protected terminals, enter:

```
TSS PER(acid) TERM(*ALL*)
```

In the last example, TERM(*ALL*) must have been previously assigned to the MSCA:

```
TSS ADD(MSCA) TERM(*ALL*)
```

Protecting Terminals Through Ownership

VM-controlled physical and logical terminals and VTAM terminals can be defined so that access to them must be explicitly authorized. Online terminals can be owned through a TSS CREATE/ADDTO TERMINAL entry. A TSS PERMIT TERMINAL authorizes both full and restricted access for terminals. To identify an online terminal to CA Top Secret, use a terminal address.

Protecting From DIAL Access

Selected virtual machines can be protected from unauthorized DIAL access. For example:

```
TSS ADDTO(SECDEPT) VMDIAL(MVS)
```

This command makes SECDEPT the owner of dial capability to the the MVS machine. That is, any user who tries to DIAL the MVS virtual machine is challenged. The user, in order to DIAL MVS, must supply his correct ACID and password to do so.

Before the user can DIAL MVS, the user must first have been PERMITted to DIAL the MVS virtual machine:

```
TSS PERMIT(USER01) VMDIAL(MVS)
```

Additionally, you may specify to which line in the virtual machine the user may dial. This is done by appending the line number to the end of the VMDIAL operand. For example, the following forces the user to dial into the MVS machine's virtual GRAF devices 0020 through 002F. An attempt to dial into any other line fails.

```
TSS PER(USER01) VMDIAL(MVS.002(G))
```

Note: The Security Validation Algorithm treats VMDIAL resources differently depending on whether or not the user includes a specific line on the dial command. If the user does specify a line ('D MVS 0200'), CA Top Secret interprets the resource requested to be VMDIAL(MVS 0020). To be allowed accesss to this resource, a user should be permitted to it either explicitly or by a generic permit, as in the following examples:

```
TSS PERMIT(USER02) VMDIAL(MVS.0020)
TSS PERMIT(USER02) VMDIAL(MVS.002(G))
TSS PERMIT(USER02) VMDIAL(MVS(G))
```

Note that because VMDIAL is a NONGENERIC resource, a permit to VMDIAL(MVS) will not allow access to VMDIAL(MVS.0020). By contrast, if the user does not include a line number in the dial command ('D MVS'), then the requested resource would grant access to this resource.

Administrators should exercise great care in defining permissions to a VMDIAL resource because of the different ways in which a user can issue a dial command.

Restricting Logons to Specific Terminals

Security Administrators force users to log on from specific terminals and with special operator cards. This option is implemented through the SOURCE keyword. For example, to restrict USER01 to logging on only from a terminal whose id is GRAF002 enter:

```
TSS ADDTO(USER01) SOURCE(GRAF002)
```

The SOURCE keyword can also be used to force certain VM ACIDs to be autologged. For example,

```
TSS ADDTO(USERENG1) SOURCE(AUTOLOG)
```

tells CA Top Secret that USERENG1 can only be initiated by a CP AUTOLOG--not by a normal user logon.

Requiring Special User Authentication

CA Top Secret supports the physical identification of terminal users through operator identification cards. This option is available for users who have IBM 3270-compatible terminals. It is implemented by attaching the OIDCARD attribute into a user's Security Record. For example, the following tells CA Top Secret to prompt for the operator id card to be inserted into the terminal's badge reader whenever USER01 logs on to a virtual machine.

```
TSS ADDTO(USER01) OIDCARD
```

Controlling Access to Virtual Machines

CA Top Secret allows an authorized ACID use of a virtual machine. This may be accomplished by the VMMACH (VM Machine) resource.

There are five applications of the VMMACH resource:

- Alternate ACID Logon
- Group Machine Logon
- CP AUTOLOG Command
- Surrogate Processing
- APPC Connect-Time Security

Note: The use of VMMACH for alternate ACID logon, Group Machine logon, CP AUTOLOG, surrogate processing, and APPC connect-time security is protected by default in all modes. Default protection cannot be overridden by the RDT or by mode.

VMMACH Authorization for Alternate ACID Logon

By default, when a user logs on to VM, an ACID is created based upon the name of the virtual machine. The ACID which the virtual machine runs under can be overridden at logon by use of the "ACID=" parameter in the CP LOGON command.

```
LOGON userid ACID=alternate-acid
```

In order for the ACID specified to log on to the virtual machine, it must first be permitted use of that virtual machine.

```
TSS PERMIT(alternate-acid) VMMACH(userid) ACCESS(LOGON)
```

The password which is verified at logon is that of the alternate ACID. Once the logon is authorized, the virtual machine assumes all of the authority of the alternate ACID.

By using VMMACH, each user of the machine could be set up with a unique ACID and password without having to be defined in the VM directory. All actions and/or violations logged for the user by CA Top Secret reflect the alternate ACID, as well as the userid logged on, thereby maintaining individual accountability.

Group Machine Logon

Group Machine Logon allows a user to log on to a virtual machine other than the user's default virtual machine, without affecting the target machine's resource access authorizations.

Group logon is invoked through the use of the "GRPUSER=" parameter of the CP logon command. The format is:

```
LOGON userid GRPUSER=group-acid
```

The term userid refers to the target virtual machine being logged on. The term, group-acid or GRPUSER, denotes the CA Top Secret ACID of the user requesting access to the machine.

In contrast with Alternate ACID Logon, Group Machine Logon does **NOT** cause the target virtual machine to inherit the ACID of the user logging on. Instead, the group user's ACID is used for password verification and system validation only. Once logged on, the virtual machine assumes (for resource access verification purposes) its own default ACID or the ACID specified by the "ACID=" parameter in the CP LOGON command.

Authorization for Group Machine Logon requires that the group user be granted GRPLOGON access to the target virtual machine.

For example, with the following command:

```
LOGON VIRTSYS1 GRPUSER=JOEL
```

JOEL will be logged on to virtual machine "VIRTSYS1" after correctly entering his own LOGON password. Once the LOGON is authorized, the virtual machine will run under the authorities defined in ACID VIRTSYS1.

To permit this request the following permission is required:

```
TSS  PER(JOEL) VMMACH(VIRTSYS1) ACCESS(GRPLOGON)
```

Group Machine Logon may also be used in combination with Alternate ACID Logon, provided the group user has access to the ACID specified via the ACID= keyword, as well as GRPLOGON or LOGON access to the target virtual machine. For instance, with the following command, user JOEL attempts logon to target virtual machine "VIRTSYS1" and to assume the resource access authorizations of ACID "SYSPACID".

```
LOGON VIRTSYS1 GRPUSER=JOEL ACID=SYSPACID
```

The following permissions are required to grant this request:

```
TSS PERMIT(JOEL) ACID(SYSPACID)
TSS PERMIT(JOEL) VMMACH(VIRTSYS1) ACCESS(GRPLOGON)
```

VMMACH and ACID Authorization for CP AUTOLOG Command

Another application of the VMMACH resource provides authorization for the CP AUTOLOG command. Normally, when a user attempts to autolog another virtual machine the user is prompted for that virtual machine's directory password. By use of the VMMACH resource, the user issuing the AUTOLOG command does not require a password. For example, the following gives the ACID specified the ability to autolog any virtual machine whose first six characters are CMSBAT (CMSBATCH, CMSBAT1, etc.).

```
TSS PER(USER01) VMMACH(CMSBAT) ACCESS(AUTOLOG)
```

The issuer of the AUTOLOG command still needs to be authorized to use the CP AUTOLOG command. This authorization is dependent upon mode and is accomplished either by having directory class A or B, or by permission to use the CP AUTOLOG command (if the command is protected by CA Top Secret).

By using the VMMACH resource in conjunction with permission to issue the CP AUTOLOG command, an administrator can give what is normally a "class G" user the ability to autolog selected virtual machines.

To allow USER01 to autolog any virtual machine, the administrator enters:

```
TSS PERMIT(USER01) CPCMD(AUTOLOG)
TSS ADDTO(DEPT01) VMMACH(*ALL*)
TSS PERMIT(USER01) VMMACH(*ALL*) ACCESS(AUTOLOG)
```

Here are some examples of CPCMD and VMMACH:

```
TSS PERMIT(USER01) CPCMD(AUTOLOG)
TSS PERMIT(USER01) VMMAC(USER02,USER03) ACCESS(AUTOLOG)
TSS PERMIT(USER01) VMMAC(*ALL*) ACCESS(NONE)
```

The first permit gives acid USER01 the ability to issue the CP AUTOLOG command. The second allows the user to AUTOLOG virtual machines USER02 and USER03 without being prompted for the directory password. The third permit fails any attempt to AUTOLOG a virtual machine other than USER02 and USER03.

Another facility provided by CA Top Secret is that of alternate ACID support for AUTOLOG. By specifying the ACID to be used as part of the AUTOLOG command, the issuer of the command can cause a disconnected virtual machine to run under the authorities of a specific ACID. The format of the AUTOLOG command is as follows:

```
CP AUTOLOG userid ACID=alternate-acid
```

In addition to the authorities required for AUTOLOG, the issuer of the command requires authorization to the alternate acid.

```
TSS PER(autologger) ACID(alternate-acid)
```

VMMACH and ACID Authorization for Surrogate Processing

VMMACH and ACID resources also control the surrogate function.

The surrogate function in CA Top Secret offers additional flexibility in extending normal security specifications to all virtual machines.

Surrogacy provides a mechanism by which an administrator may designate authorities of specific ACIDs to virtual machines. In CA Top Secret, a virtual machine can assume the authorities of an alternate ACID -- temporarily overriding the authorities which have normally been established at logon for the virtual machine. For example, a VM batch master control machine may activate a worker machine called WRK7 with USERA's ACID when USERA requires that machine for job execution. Upon job completion, WRK7 is either reset to its original ACID or set to another submitter's ACID by the batch master control machine. At all times USERA, if logged on, always retains access to its resources.

By supporting the use of alternate ACIDs for virtual machines, CA Top Secret keeps your data secure whenever tasks are assigned to worker machines. Security violations can be traced to either the submitting ACID or the worker machine.

See the *Customization Guide* for a description of the surrogate interface.

Surrogate Control

Access verification for surrogate control is automatically invoked when certain program products use the VM Alternate Userid Interface (Diagnose X'D4'). Refer to the *Customization Guide* for details about this interfaces.

For your convenience, for applications written in REXX or EXEC, or where Assembly language is inappropriate, a CMS module called SUROGATE is provided on the installation tape. It implements the surrogate user facility through command forms. This module can be found on the appropriate test or production minidisk of the CA Top Secret product maintenance virtual machine.

Surrogacy is administered through the following command functions:

```
SUROGATE SET userid|* [ acid ]
SUROGATE RESET userid|*
```

SET

Establishes an Alternate ACID under which a virtual machine runs.

Permission for use of the virtual machine and authorization to the Alternate ACID being set are granted as follows:

```
TSS PERMIT(issuer) VMMACH(target-of-set) ACCESS(SUROGATE)
TSS PERMIT(issuer) ACID(acid-used-in-set)
```

- *issuer*—Is the controller ACID
- *target-of-set*—Is the worker virtual machine (CP directory id). It must be logged on when SUROGATE SET is issued
- *acid-used-in-set*—Is the Alternate ACID used in SUROGATE SET or RESET.

Note: The ACID authority is not required if issuer has the NOSUBCHK attribute.

SUROGATE SET remains in effect until it is explicitly reset bySUROGATE RESET or until the target of the set logs off.

VMMACH access validation in surrogate processing is always treated in FAIL mode.

Userid

Gives the id of the virtual machine that acts with another's authority.

*****

Indicates the vmid of the issuer authorized as a surrogate controller and is also the default.

Acid

Is the Alternate ACID providing the appropriate authorities.

RESET

Removes the Alternate ACID established by a prior SUROGATE SET. Permission for use of the virtual machine and authorization to the Alternate ACID being set must be in effect (see the TSS PERMIT commands shown in the SET explanation).

In the example shown below, a VM master control machine (VMMASTER) has been permitted the proper authority for:

- The surrogate user function diagnose code (D4)
- Surrogate access to a worker machine (WRK7)
- Access to an ACID (USERA) whose authority is to be delegated.

The SUROGATE SET designates USERA's authorities to WRK7.

```
TSS PERMIT(VMMASTER) DIAG(D4)
        . . .
TSS PERMIT(VMMASTER) VMMACH(WRK7) ACCESS(SUROGATE)
TSS PERMIT(VMMASTER) ACID(USERA)
        . . .
CP AUTOLOG WRK7
SUROGATE SET WRK7 USERA
        . . .
```

VMMACH Authorization for APPC Connect-Time Security

The VMMACH resource also controls APPC connect-time security on local APPC connection requests.

Permission for an ACID to connect to another virtual machine via APPC is granted as follows:

```
TSS PERMIT(acid) VMMACH(target) ACCESS(APPC)
```

Note: In direct response to client requests, this additional APPC security has now been made optional. System security performance will be improved by turning off this feature for those clients that have no need for APPC security.

CA Top Secret is shipped with the APPC security calls disabled. To enable the security, add OPTIONS(1) to start up the Parameter File.

Support for OpenExtension VM

Release 1.5 offers security for the OpenExtension VM environment. Specifically, CA Top Secret supports the following services in an OpenExtension VM environment:

- Callable services
- Byte File System (BFS)

- User ID (UID) and Group ID (GID) definitions
- Audit records for OpenExtension VM
- Shell Setup Utility

CA Top Secret provides the following features to secure an OpenExtension VM environment:

- Ability to create ACIDs needed to install and control access to OpenExtension VM and the Byte File System (BFS).
- CA Top Secret records for OpenExtension VM.
- Ability to log and report on OpenExtension VM security calls.

Assigning UIDs and GIDs

In the OpenExtension VM environment, every User ACID must be assigned a User ID (UID) and every Group ACID a Group ID (GID). UIDs and GIDs can be any numeric value from 0 through 2,147,483,647.

Note: The ACIDs used by the Shared File System (SFS) server machines must have UID(0).

Defining OpenExtension VM Groups

OpenExtension security is based on user and group ownership of files and processes. CA Top Secret uses the GROUP type ACID to assign users to an OpenExtension VM group.

When group access checks are performed, CA Top Secret compares the GID of the file to the GIDs of all the groups defined to the ACID. If a match is found, CA Top Secret uses GROUP permissions to determine the user's access to the file.

Controlling Access to the Byte File System

The Byte File System (BFS) is a tree-structured file system consisting of directories and files. Security for the file system directories and files is based on a UNIX model of security. Each file and directory is assigned an owning UID and an owning GID.

The HOME keyword defines the initial directory pathname. This is the initial directory used when a user enters the OPENVM command or enters the OVM shell.

CA Top Secret Records for OpenExtension VM

This section describes the CA Top Secret records for OpenExtension VM.

ACID Records

OpenExtension VM UIDs are defined to CA Top Secret by the OMVS segment in the ACID Record. The OMVS segment of the ACID Record contains four keywords: ROOT, UID, HOME, and OMVSPGM.

The OMVSPGM keyword defines the user's OpenExtension VM shell program. This is the first program started when the OPENVM command is entered or when an OpenExtension VM batch job is started.

GROUP Profile Records

OpenExtension VM groups are defined to CA Top Secret by the GROUP type ACID. The GROUP type ACID contains the OMVS segment and the GID keyword.

POSIXMGRP Control Option

This control option must be set to enable ESM security for OE/VM. For further information, see the *Control Options Guide*.

Applicable Keywords

The following CA Top Secret keywords are used in support of the IBM OpenExtension VM environment.

- DFLTGRP
- GROUP
- GID
- HOME
- OMVSPGM
- UID
- ROOT

Using the Secured Signon Feature

CA Top Secret supports the Secured Signon feature of the IBM Network Security Program (NETSP) that must use the NETSP keyword. This feature lets network users from one or more nodes communicate with a host using a PassTicket. A PassTicket is a dynamically generated, one-time-only, password substitute with a limited lifespan. Using PassTickets eliminates the need to send an ACID's password across the network when signing on to other nodes, and allows CA Top Secret to validate the user by checking the generated PassTicket. PassTickets can be used to access any facility where a password is used to sign on.

How to Set Up the Environment

All systems using PassTickets must have identical application names and session keys for all nodes on the network. For example, if you are accessing on VM from an OS2 system that supports PassTicket processing, the application name **must** match the VM SYSID. In addition, all systems must use the same standard rules to establish the validity of the PassTicket. Check your IBM documentation if you are unfamiliar with these rules.

Support for Multiple Console System (MCS) Commands

CA Top Secret supports the TSO/E Extended MCS Console Facility. The Multiple Console System (MCS) allows you to define multiple consoles, enter console commands, and receive console messages from various terminals defined as remote consoles. Full administration can be performed from CA Top Secret.

Applicable Keywords

CA Top Secret uses the following keywords to maintain MCS attributes. These keywords allow an ACID to specify and change MCS attributes. These keywords can also be extracted on MVS using the RACROUTE TYPE=EXTRACT call command using a prefix of MCS instead of OPER (such as MCSALTG instead of OPERALTG).

- MCSALTG
- MCSAUTH
- MCSAUTO
- MCSCMDS
- MCSDOM
- MCSKEY
- MCSLEVL

- MCSLOGC
- MCSMFRM
- MCSMGID
- MCSMON
- MCSROUT
- MCSSTOR
- MCSUD

How to Define PassTickets

To define PassTickets to CA Top Secret you must identify each application that can accept a PassTicket and assign that application a unique password called a session key. This information is added to the Node Descriptor Table (NDT) using the PSTKAPPL (PassTicket application) and SESSKEY (session key) parameters of the TSS ADD command function. Both parameters must be supplied. Refer to the *Command Functions Guide* for detailed information on these keywords.

This example assigns session key 296LFD to an application named PRIDEV02.

```
TSS ADD(NDT) PSTKAPPL(PRIDEV02) SESSKEY(296LFD)
```

The Node Descriptor Table (NDT) contains all PassTicket application and session key-related node information. The NDT is a global record similar to the Resource Descriptor and Field Definition Tables.

Chapter 4: Ownable Resource Security

This section contains the following topics:

Resources

Most resources must first be owned before their use can be authorized. The principal exceptions are commands, which are protected through the USERx class resources. Either the TSS CREATE or ADDTO functions must be used to assign ownership of any of the ownable resources. The former is used at the time an ACID is being defined. The latter can be used at any time.

The administrative authority required to add or remove ownership of resources is RES(OWN). The keyword RESOURCE applies to all ownable resource classes. Specific resource class keywords may be used instead of RESOURCE to restrict the authority of the administrator. For example, PGM(OWN) authorizes the administrator only to add or remove ownership of programs. The ACID to which the resource is being added or removed must lie within the scope of the administrator.

“Default protection of resources” means that an ownable resource will have security protection even if it is not defined to CA Top Secret. That is, a security violation will occur if a request is made to access any unowned resource. (In FAIL MODE this situation exists automatically for data sets.)

To extend default protection to all resources, pre-defined in the RDT Record, whether the governing mode is WARN, IMPLEMENT, or FAIL, assign the DEFPROT attribute to the specifically named resource class which is described in the subsequent headings.

An administrator can give default protection to any pre-defined resource.

The Resource Descriptor Table (RDT) Record

The Resource Descriptor Table (RDT) is a special ACID, like AUDIT and ALL, that is used to define resource classes and their properties.

The RDT contains pre-defined CA Top Secret resource classes (VM, MVS, and VSE resources), including resources used and reserved by other Computer Associates product interfaces. In addition, the RDT can contain dynamically defined resource classes. The user can redefine access levels to these resource classes in more meaningful ways, such as foreign language words for READ and UPDATE, etc.

For managing the contents of the RDT Record, the administrator can specify attributes, access levels, and default access levels through the TSS command.

Modifying the RDT

The user may have one or several reasons for updating or modifying the RDT. These reasons can be divided into two main categories:

Modifying Existing Resource Classes

The user may want to modify an existing resource class to change its attributes. For example, the user can add default protection by assigning the DEFPROT attribute to the appropriate TSS command function. DEFPROT and the other attributes are discussed later in this chapter.

Note: For pre-defined resource classes only EXIT, DEFPROT, MASK, and MERGE attributes can be used.

Defining New Resource Classes

The user may want to define non-CA Top Secret resource classes in the following situations:

- Users may be instructed to define a new resource class required by a Computer Associates or other vendor-supplied product.
- Users must also define resource classes for security calls issued by installation-written routines with their own resource classes, and any other customized site applications with resources and security of their own.

Implementing the RDT

This section describes how to implement the RDT record.

Command Syntax

The following TSS command functions that relate to the RDT are shown below. For complete information, refer to the Command Functions Guide.

To define a new resource class to the RDT, enter:

```
TSS ADDTO(RDT) RESCLASS(resource class name) RESCODE(hex code)
   [ATTR(attribute list)] [ACLST(access level list)]
   [DEFACC(default access level)]
```

To change the values of previously-defined resource classes, enter:

```
TSS REPLACE(RDT) RESCLASS(resource class name)
   [ATTR(attribute list)] [ACLST(access level list)]
   [DEFACC(default access level)]
```

To remove a previously-defined resource class definition, enter:

```
TSS REMOVE(RDT)  RESCLASS(resource class name)
```

To list resource classes, enter:

```
TSS LIST(RDT) [RESCLASS(resource class name(s)]
```

The two keywords RESCLASS and RESCODE are required when adding a resource to the RDT Record. Only RESCLASS is required for modifying, listing and removing a particular resource class name. Examples using these keywords are provided later in this chapter.

RESCLASS

Is the eight-character resource class name. The TSS command, logging, and the security interface honor this name.

RESCODE

Is the two-digit hexadecimal code which is used internally by CA Top Secret to identify the resources of this particular class. This identifier is used in trace as well as logging information. When adding a resource class to the RDT Record, you can use any hexadecimal code between 001 - 03F and 101 - 13F which is reserved for dynamically-defined RESCODEs.

Optional keywords that can be used when dynamically defining a resource to the RDT Record or when modifying an existing resource class follow.

ATTR

Contains one or more of the following operands:

- EXIT—Calls the installation exit for this resource class
- NOEXIT—Deactivates the installation exit call
- DEFPROT—Protects this resource class by default
- NODEFPROT—Deactivates default protection
- PRIVPGM—Supports PRIVPGM for this resource class
- NOPRIVPGM—Deactivates PRIVPGM support
- GENERIC—Supports generic prefixing for this resource class
- NONGENERIC—Deactivates generic prefixing and treats the resource names as fully qualified names
- LIB—Supports LIB with PRIVPGM for this resource class
- NOLIB—Deactivates support for LIB with PRIVPGM
- LONG—Supports 44 character entities for this resource class
- SHORT—Deactivates 44 character support
- MERGE—Uuses AUTH(MERG) for access checking of this resource class
- NOMERGE—Deactivates AUTH(MERG) option for access checking
- ALLMERGE—Uses AUTH(ALLMERGE) for access checking of this resource class
- NOALLMERGE—Deactivates AUTH(ALLMERGE) option for access checking
- VMUSER—Supports VMUSER for this resource class
- NOVMUSER—Deactivates VMUSER support

Note: For PRIVPGM, LIB and VMUSER the security driver must also support these features. For all pre-defined CA Top Secret resource classes (such as DATASET and VOLUME), only the DEFPROT, EXIT, and MERGE attributes may be altered through the TSS REPLACE(RDT) command function.

Consider the following examples. An administrator wants to add PRODUCTA to the RDT Record, and give it default protection. He enters:

```
TSS ADDTO(RDT) RESCLASS(PRODUCTA) RESCODE(10) ATTR(DEFPROT)
```

If an administrator wishes to remove an attribute that he has assigned to a specific resource class, he simply uses the TSS REPL(RDT) command function and prefixes the attribute with NO. For example:

```
TSS REPLACE(RDT) RESCLASS(PRODUCTA) ATTR(NODEFPROT)
```

In this example, the resource class PRODUCTA no longer has default protection.

To remove the LONG attribute which is already attached to a specific resource class, specify ATTR(SHORT). This is the only exception that does not use the prefix NO when when attempting to remove an already defined attribute.

You can remove the LONG attribute only with dynamically defined resources, but not pre-defined resources. To list data concerning how PRODUCTA will be processed, the administrator enters:

```
TSS LIST(RDT) RESCLASS(PRODUCTA)
```

If an administrator wants to remove PRODUCTA from the RDT, he enters:

```
TSS REMOVE(RDT) RESCLASS(PRODUCTA)
```

Note: Removing a resource class from the RDT requires that all owned resources belonging to that particular resource class were previously removed.

ACLST

Lists up to 20 access levels for this resource class. If not specified, then the resource class does not support access level checking. It is recommended that ALL, CONTROL, UPDATE, and READ are defined. If the CA Top Secret defined access levels are used, the administrator can simply specify a list.

For example:

```
ACLST(READ,WRITE)
```

However, if he wants his own unique access levels, he must specify the hexadecimal values associated with each access level.

For example:

```
ACLST(ABC=0500,XYZ=0600,...)
```

The administrator can also mix defined with his own unique access levels.

For example:

```
ACLST(READ,XYZ=0600)
```

The access level list is supported both by the TSS command during administration, access validation, and for logging and reporting.

CA Top Secret pre-defined access levels follow and are given in hexadecimal values:

- NONE(0000)
- SET(0040)
- INQUIRE(0080)
- NOCREATE(0100)
- PURGE(0100)
- FEOV(0200)
- BROWSE(0200)
- CONTROL(0400)
- MULTI(0400)

- SCRTCH(0800)
- REPL(0800)
- CREATE(1000)
- DELETE(1000)
- GRPLOGON(1000)
- FIND(1000)
- SUROGATE(2000)
- WRITE(2000)
- MWRITE(2400)
- READ(4000)
- AUTOLOG(4000)
- MREAD(4400)
- FETCH(8000)
- UPDATE(8000)
- BLP(8000)
- ALL(FFFF)
- LOGON(8000)

Note: ALL and NONE cannot be coded.

If an administrator wishes to remove an access level list, he simply enters:

```
TSS REPLACE(RDT) RESCLASS(PRODUCTA) ATTR(NOACCESS)
```

DEFACC

Sets the default allowed access for this resource. If not specified, the default access is NONE.

Maintaining the RDT

To maintain and list the RDT Record, the administrator needs the following:

```
TSS ADMIN(acid) MISC1(RDT)
```

The administrator now wants to add the new resource class to the RDT Record with READ and WRITE access levels. To dynamically add this new resource class, he enters:

```
TSS ADDTO(RDT) RESCL(PAY) RESCODE(12) ACLST(WRITE,READ)
```

In order to give the administrator appropriate authority to assign ownership and grant users access to a new resource class PAY, he needs the following:

```
TSS ADMIN(acid) PAY(OWN,XAUTH) ACC(READ,WRITE)
```

The administrator can now add this new resource class name PAY to a department that is within his scope. He enters:

```
TSS ADDTO(DEPT01)  PAY(401K)
```

Note: 401K is a resource name that is now accessed by the resource class PAY.

To permit this resource to USER01 with READ access, the administrator enters the following TSS command:

```
TSS PER(USER01) PAY(401K)  ACC(READ)
```

To list the data concerning PAY, the administrator enters:

```
TSS LIST(RDT) RESCLASS(PAY)
```

If the administrator eventually wants to display the owner(s) of the resource name 401K, he simply enters:

```
TSS WHOOWNS PAY(401K)
```

Likewise, if the administrator would like to display the authorization(s) for the resource class name, he would enter:

```
TSS WHOHAS PAY(401K)
```

Using 2-Byte Resource Codes

An additional 256 resource classes are represented internally with 2-byte resource codes. These resource classes are incompatible with lower-level releases, which only allow 1-byte resource codes. Resources with 2-byte resource codes should not be activated in any system until all systems sharing the Security File are running with CA Top Secret 1.5 or above.

Before 2-byte resource codes can be added to the RDT, the control option RDT2BYTE must be set. Once a resource class has been added to the RDT with a 2-byte RESCODE parameter, the RDT2BYTE is said to have been activated and irrevocable changes have been made to the Security File. In this case, CA-Top Secret 1.5 will initialize with the following message:

```
TSS9053I RDT2BYTE OPTION ACTIVATED
```

If RDT2BYTE has been activated, any CA Top Secret 1.4 system attempting to initialize with such a Security File will terminate with the message:

```
TSS9996E Security File contains two byte rescodes and cannot be used by
this release
```

If a CA Top Secret 1.4 system shares a Security file with a CA Top Secret 1.5 system and the latter defines a 2-byte resource code resource class to the RDT, results can be unpredictable and may compromise Security File integrity. When the 1.4 system is shut down, it will no longer be able to initialize with the same Security File.

Unlike other control options, once a RDT2BYTE control option has been activated, it cannot be modified by altering the Parameter File or executing a MODIFY command.

After a Security File has been modified with a two-byte RESCODE resource, that file cannot be used by CA Top Secret 1.4. If your Security File has this change but you wish to go back to CA Top Secret 1.4, you must restore a backup of the Security File that never contained a resource with a 2-byte resource cod Then run TSSRECVR to rebuild a Security File that is compatible with CA Top Secret 1.4.

Field Descriptor Table (FDT) Record

The Field Descriptor Table is a reserved or special ACID that contains and manipulates fields. These fields give you the ability to define variable length data to the Security File, and can be predefined by Computer Associates or user-defined. Before attaching a field to an ACID, it must first be defined to the FDT. Predefined fields can be modified immediately since they already exist. A sample of some of the predefined fields you can find in the FDT appear below.

- LANGUAGE—PCADMIN
- LTIME—PCDSDAYS
- OPCLASS—SMSAPPL
- OPIDENT—SMSDATA
- OPPRTY—SMSMGMT

Modifying the FDT

You can have one or more reasons for updating or modifying the FDT. These reasons can be divided into two main categories:

- Modifying Existing Fields

  You want to modify an existing field to change its characteristics. For example, if you want to change the displayed title of a field shown when listing a user, you would enter:

  ```
  TSS REP(FDT) FDTNAME(FIELD1) DISPLAY('USER FIELD1')
  ```

- Defining New Fields

  You want to use fields to define installation information (by user ACID) that can be manipulated or extracted by application programs. Applications can extract and update this information using the Application Interface.

  Once defined, you can add these fields to ACIDs using the TSS commands. You can ADD, REMOVE, REPLACE, or LIST the defined data the same way you would with fields predefined by CA Top Secret.

  You can add these fields to a profile ACID to create a role based profile that can be added to any user. The extract process returns field data from the first connected profile if the field is not found in the user record of the ACID. Use the FIRST keyword when adding the new profile to an ACID to ensure it is the first profile/group found in the list.

Note: The Security File is structured to provide CA Top Secret with the best performance possible. However, a wide use of user-defined data can cause a situation where the original CA Top Secret information becomes a fraction of the total Security File causing an unbalanced I/O as well as a misuse of storage and CPU. You should re-examine the Security File size according to the amount of user data that the file is expected to hold.

Maintaining the FDT

The TSS commands that relate to the FDT appear in the following sections. For complete information, refer to the Command Functions Guide.

Administrative Authority

You must have MISC1(RDT) authority to ADD, REMOVE, or REPLACE fields in the FDT Record.

Defining New Fields to the FDT

You can define a new field to the FDT by using the command:

```
TSS ADD(FDT) FDTNAME(field-name)
          FDTCODE(hex-code)
          SEGMENT(segment-name)
          MAXLEN(nn)
          DISPLAY(display-name)
          [ATTR(MIXED)][(NONDISP)]
```

FDTNAME

Adds an up to eight-character user-defined field to the FDT Record when FDT is the target ACID name, and allows one field-name per command that can be a letter, number, or special character.

FDTCODE

Adds a user-defined hex-code to the FDT Record when FDT is the target name, and allows one hex-code per command that can range from 01 to FF.

SEGMENT

Assigns an up to eight-character field to a specific segment, and allows one segment-name per command that can be a letter, number, or special character. You cannot add user-defined fields to predefined segments. The following are predefined CA Top Secret segments:

- ALL—LANGUAGE
- BASE—OMVS
- CA-PC—OPERPARM
- CICS—TSO
- DFP—WORKATTR
- DLFDATA—IMS

MAXLEN

Defines the number of bytes that can be entered for the user-defined FDT entries, and allows one MAXLEN value per command. The total of all user-defined fields should not exceed 32,767 bytes.

DISPLAY

Defines an up to 11-character display-name with its associated defined field and segment in the FDT Record, and allows one display-name per command that can be a letter, number, or special character. The display-name must be enclosed in single quotes if it contains blanks.

ATTR

Displays fields from the FDT in mixed case format when ysing the MIXED attribute. If you do not specify ATTR, it defaults to uppercase.

If a field is defined with the NONDISP attribute, the field cannot be seen by an administrator using a TSS LIST function on the ACID. The data can be extracted and/or modified as normal using the Application Interface or the RACROUTE macro.

You may want to use non-CA Top Secret fields to define installation information (by user ACID) that can be maintained or extracted by application programs. Applications can extract and update this user information using the RACROUTE macro.

Extract of the FDT is performed from the USER record of an ACID or from the first profile connected to an ACID. This allows the administrator to assign security related fields to a role based profile that can be added to any user. Use the FIRST keyword when adding the new profile to an ACID to ensure it is the first profile/group found in the list. The profile is searched for a field if it is not found in the USER record for the ACID.

Examples

This example creates an area code and home phone field called HPHONE that will belong to a segment named HPHNATTR and a display called HPHNUM. The maximum length for the field is 12 bytes and has a hex-code of 01. This field is created by entering the command shown below.

```
TSS ADD(FDT) FDTNAME(HPHONE) FDTCODE(01)
   SEGMENT(HPHNATTR) MAXLEN(12) DISPLAY(HPHNUM)
```

To add the contents of this area code and home phone field to its respective owner, enter:

```
TSS ADD(USER01) HPHONE('908 780 5550')
```

Note: Use single quotes when the field contains blanks.

Changing User-Defined Fields in the FDT

You can change the values of user-defined fields by entering the TSS REPLACE command function shown below.

```
TSS REPLACE(FDT) FDTNAME(field-name)
   SEGMENT(segment-name) MAXLEN(nn) DISPLAY(display-name)
```

FDTNAME

Contains an up to eight-character user-defined field in the FDT Record whose values can be removed or changed using the TSS REPLACE command function, and allows one field-name per command that can be a letter, number, or special character. You must use at least one one of the keywords that are described on the following page to modify a value of a user-defined field.

SEGMENT

Changes a specific segment that contains a group of fields. Specify one segment-name per command, that can be a letter, number, or special character. You cannot modify user-defined fields to predefined segments. For a list of the predefined CA Top Secret segments, refer to Defining New Fields to the FDT.

MAXLEN

Changes the number of bytes that can be used for the user-defined field, and allows one MAXLEN value per command. The total of all user-defined fields should not exceed 32,767 bytes.

DISPLAY

Changes an up to 11-character display-name that is associated with a particular field and segment in the FDT Record, and allows one display-name per command that can be a letter, number, or special character. The display-name should be enclosed in single quotes if it contains blanks.

Example

To change the maximum length for the HPHONE field from 12 to 14 bytes by entering the command shown below.

```
TSS REP(FDT) FDTNAME(HPHONE) MAXLEN(14)
```

Removing Fields From the FDT

You can remove a user-defined field by entering the TSS REMOVE command function shown below.

```
TSS REMOVE(FDT) FDTNAME(field-name)
```

FDTNAME

Removes an up to eight-character user-defined field from the FDT Record when FDT is the target ACID name, and allows one field-name per command that can be a letter, number, or special character. Only the FDTNAME is required for removing a user-defined field.

For example, to remove the HPHONE field from the FDT, enter the command shown below.

```
TSS REMOVE(FDT) FDTNAME(HPHONE)
```

Listing the FDT

This section describes how to list the FDT.

Listing the Entire Record

The following TSS LIST command function lists the entire FDT including both predefined and user-defined fields.

```
TSS LIST(FDT)
```

Listing a Field

The following TSS LIST command function lists a specific field.

```
TSS LIST(FDT) FDTNAME(field-name)
```

FDTNAME

Lists an up to eight-character field with its associated display and segment in the FDT Record, and allows one field-name per command that can be a letter, number, or special character.

For example, you can list the HPHONE field that also gives you its associated display and segment by entering the command shown below.

```
TSS LIST(FDT) FDTNAME(HPHONE)
```

Listing a Field Code

The following TSS LIST command function lists a specific field code which must be a user-defined field.

```
TSS LIST(FDT) FDTCODE(hex-code)
```

FDTCODE

Lists a user-defined field code in the FDT Record when FDT is the target ACID name, and allows one hex-code per command with a value ranging from 01 to FF.

For example, you can list hex-code 01 by entering the command shown below.

```
TSS LIST(FDT) FDTCODE(01)
```

Listing a Segment

The following TSS LIST command function lists a specific segment-name and its associated fields and displays.

```
TSS LIST(FDT) SEGMENT(segment-name)
```

SEGMENT

Lists an up to eight-character segment-name with its associated field and display in the FDT Record, and allows one segment-name per command, that can be a letter, number, or special character.

For example, to list the segment HPHNATTR to which the home area code and phone number belong, enter the command shown below.

```
TSS LIST(FDT) SEGMENT(HPHNATTR)
```

Listing a Display

The following TSS LIST command function lists a specific display.

```
TSS LIST(FDT) DISPLAY(display-name)
```

DISPLAY

Lists up to an 11-character display-name with its associated field and segment in the FDT Record, and allows one display-name per command that can be a letter, number, or special character.

For example, to list the HPHNUM field to which the home area code and phone number belong, enter the command shown below.

```
TSS LIST(FDT) DISPLAY(HPHNUM)
```

You can limit the authority of an ACID to only listing the FDT, by assigning the MISC8(LISTRDT) authority.

Selecting Resource Owners

Resources may typically be owned by departments, divisions or zones and then PERMITted to users and profiles on an as-needed basis. Following this approach makes security administration simpler and more effective. Reasonable exceptions to the “rule” might be to let users own their minidisks or those OS/DOS data set files with names that begin with the user’s id.

The owner of a resource automatically has unlimited access to that resource. On the other hand, a user who has been PERMITted to use a resource might have unlimited or limited access to the resource.

Zone, division, department, and user ACIDs can all own resources. Of these ACID types, however, only users and profiles can be authorized (with TSS PERMIT) to access resources.

Note: If a particular department is given ownership of many resources that are PERMITted many times (500+), then it is a good idea to create some dummy departments and split up the ownership of these resources among them. This enhances processing efficiency by achieving a more balanced distribution of information on the CA Top Secret Security File.

Using Generic Prefixing

Resources are defined to CA Top Secret either by their full name or through a generic prefix, which allows a group of similar resources to be defined to CA Top Secret simultaneously. For all resources (except DIAGNOSE), generic prefixing can be a useful tool for Security Administrators.

By entering the TSS command shown below, the Security Administrator authorizes USER99 to use any minidisk owned by MAINT.

```
TSS ADD(USER99) VMMDISK(MAINT.)
```

Usually, the minimum length permitted for a generic prefix is one character. However, minidisks and OS data sets have a two-character minimum length. Also, most resource classes have an eight-character maximum length for prefixes. Again, minidisks and data sets do not adhere to this standard: for minidisks, up to 13 characters may be used; for OS data sets, up to 26 characters may be used to designate the generic prefix.

CA Top Secret allows identical prefixes, as long as they are used with different resource types. For example, VMRDR(MAINT) and VMMD(MAINT) do not conflict.

Using the GENERIC-NONGENERIC Attribute

A new RDT attribute called NONGENERIC has been incorporated in CA Top Secret. This new attribute causes a general resource to be treated as a fully qualified name rather than as a generic prefix.

The NONGENERIC attribute can support both long and short resource classes. This attribute does not, however, support the following resources which support masking:

- DSNAME
- DB2DBASE
- DB2TABLE
- DB2TABSP
- JESJOBS
- JESSPOOL
- OPERCMDS
- VMMDISK

As an example, an administrator can permit a user to resource IUCV(PSRV), which would allow an IUCV connection to virtual machine PSRV as well as to virtual machine PSRVTEST or any other virtual machine whose first four characters match the prefix, PSRV. If the NONGENERIC attribute is activated, however, a permit to IUCV(PSRV), only allows the user to connect to virtual machine PSRV and not PSRVTEST. In order for the user to be allowed to issue either transaction, the permit must be done to IUCV(PSRV(G)).

Note: The NONGENERIC attribute causes a general resource to be treated as a fully-qualified resource only in a permit. By contrast, ownership (add) of a resource with the NONGENERIC attribute is considered to be GENERIC, and a (G) will appear after the resource name in TSS WHOHAS, WHOOWNS, and LIST output. To have a resource with the NONGENERIC attribute owned as a fully-qualified resource, the resource name in the TSS ADD command should be surrounded with single quotes and a trailing blank, as in the following example:

```
TSS ADD(USER01) IUCV('PSRV ')
```

To alter a particular general resource class to conform to the NONGENERIC attribute, the administrator enters:

```
TSS REPLACE(RDT) RESCLASS(VMRDR) ATTR(NONGENERIC)
```

Where the VMRDR keyword is the resource class to be altered.

To remove the NONGENERIC attribute, the administrator enters:

```
TSS REPLACE(RDT) RESCLASS(VMRDR) ATTR(GENERIC)
```

When changing the resource class from GENERIC to NONGENERIC, or from NONGENERIC to GENERIC, the security validation behavior for all existing definitions is preserved. However, resources will list differently and administrative commands which follow will have a different effect.

The NONGENERIC attribute applies to the following resources by default:

- IUCV
- VMCF
- VMDIAL
- VMMACH
- VMRDR
- VMUSER

Undercutting

An undercut occurs when a generic prefix is used to establish ownership that is generically higher (more inclusive) than an existing prefix. For example, the prefix "VM" undercuts the prefix "VMTEST."

CA Top Secret does not allow an undercut unless all prefixes are owned within the scope of the administrator specifying the undercut.

Also, CA Top Secret does not allow the use of a generic prefix to establish ownership that is generically lower (less inclusive) than an existing prefix. For example, the prefix "VMTEST" cannot be used to undercut the "VM" prefix

If the UNDERCUT is valid, CA Top Secret automatically transfers ownership of the resources specified by the generically lower prefix to the new owner. See Transferring Ownership of Resources for information about undercutting resources.

Undercutting restrictions do not apply when the subsequent prefix is used to specify resource authorization rather than resource ownership. For example, the following are valid:

```
TSS ADD(DEPT01) VMRDR(BAT)
TSS PERMIT(USER02) VMRDR(BATCH)
```

Avoid assigning ownership of a prefix that overrides a previously owned prefix and inadvertently “undercutting” the existing prefix.

Determining Resource Ownership

The owner of a resource can be found through the use of the TSS WHOOWNS function. RESOURCE(INFO) administrative authority is required for its use. All resources that generically match the designated resource--and are within the scope of the administrator who issued the WHOOWNS--are displayed in alphabetical order.

All owned resources of a particular resource type may be displayed by using an asterisk (*) as the operand of the WHOOWNS. This capability is also delimited by the scope of the administrator using the WHOOWNS.

For example, to find out what diagnose codes are within your scope enter this TSS command:

```
TSS WHOOWNS DIAGNOSE(*)
```

Transferring Ownership of Resources

The ownership of a resource may be transferred from one ACID to another by using TSS ADDTO(acid). Assuming the administrator entering the transfer has the appropriate administrative authority, CA Top Secret automatically transfers ownership of the specified resources from the former to the new owner. The essential administrative authority required is RESOURCE(OWN). Also, both the new and former owners must fall within the scope of the administrator.

The TSS command entered is the same as if the resource were currently unowned with the addition of the UNDERCUT keyword as a “flag” that resource ownership is being transferred. Only the new owner is identified; no identification of the current owner is required.

To illustrate this procedure, assume that the virtual reader CMSBATCH is owned by USER01, and you want to transfer ownership to USER02. A single CA Top Secret entry accomplishes this transfer:

```
TSS ADDTO(USER02) VMRDR(CMSBATCH) UNDERCUT
```

In effect, the UNDERCUT keyword tells CA Top Secret that whoever is making this entry is aware that CMSBATCH is already owned. Note that the UNDERCUT keyword is required whenever ownership is being transferred and not just in undercutting type situations.

Whenever ownership is transferred, CA Top Secret automatically confers full access authorization to the former owner (provided that the former owner was a user or profile). Continuing with our example, the result is the same as if this entry had simultaneously been made:

```
TSS PERMIT(USER01) VMRDR(CMSBATCH)
```

Likewise, validly entering the following commands in the indicated order:

- TSS ADDTO(USER01) VMMD(USER01)
- TSS ADDTO(USER02) VMMD(USER) UNDERCUT

makes USER02 the owner of all minidisks prefixed by USER (USER01, USERA, USERXYZ). (Thus, ownership of USER01 minidisks has been “undercut”--see earlier discussion.) Furthermore, the result is as if the following had also been entered.

```
TSS PERMIT(USER01) VMMD(USER01) ACCESS(ALL)
```

NOPERMIT

To forestall the automatic granting of full access to the former owner use the NOPERMIT keyword. Refer again to the previous VMRDR(CMSBATCH) example. If the administrator had entered:

```
TSS ADD(USER02) VMRDR(CMSBATCH) UNDERCUT NOPERMIT
```

USER01 retains no access authorization to CMSBATCH.

Authorizing Resource Access

Authorization through the TSS PERMIT function allows users to use designated resources in either an unlimited or specifically restricted manner. Restrictions are specified by incorporating the appropriate keywords into the PERMIT entry. For example, the following allows USER01 read-only access to the MAINT minidisk through VM, but only on the system that has an SMFID of TS01.

```
TSS PERMIT(USER01) VMMD(MAINT) FAC(VM) SYSID(TS01)
ACCESS(READ) TIME(08,17) FOR(30)
```

Access is allowed only from 8 A.M. to 6 P.M. for a period of 30 days, commencing from the date of this entry.

Note: USER01 requires the ability to access VM through a previous CREATE/ADDTO FAC(VM) entry for the example to be valid. If no facility restriction had been specified above, then USER01 may have been able to access MAINT through any facility that is normally accessible.

For a list of all the keywords that can be associated with the PERMIT function, refer to the Command Functions Guide.

The Security Administrator needs RESOURCE(XAUTH) authority to PERMIT or REVOKE access to resources. The authority can be also be restricted to particular types of resources, such as MINIDISK(XAUTH).

Note that RES(XAUTH) is an atypical administrative authority in that it allows the administrator to assign or revoke access authorizations to any ACID--provided that the resource involved is within the administrator's scope.

Allowing Resource Access with ACTION(ADMIN)

Under normal circumstances, CA Top Secret allows an administrator to PERMIT and REVOKE only those resources which fall under his scope. When specifying ACTION(ADMIN), the administrator gives authority to another administrator requesting the ability to PERMIT and REVOKE resources outside of his scope. For example, the following allows DCA05 to permit or revoke access to this resource to other users although the resource itself is not owned within his scope.

```
TSS PER(DCA05) VMMD(MAINT.019) ACTION(ADMIN)
```

Denying Resource Access with ACTION(DENY)

A TSS PERMIT(acid) ACTION(DENY) entry may be used to selectively deny an ACID access to any resource which does not support an access level. For example, let us assume that USER01 is connected to PROF01 which provides USER01 with several resource authorizations including access to all the terminals in the Accounting Department.

Suppose USER01 is denied access to two terminals: GRAF04DC and GRAF05DC. To disallow authorization for USER01 to access these terminals, the administrator enters this command:

```
TSS PER(USER01) TERM(GRAF04DC,GRAF05DC) ACTION(DENY)
```

Whether a request by USER01 to use either of these terminals is actually failed still depends upon the MODE that applies. To ensure that this request automatically fails, the administrator enters:

```
TSS PER(USER01) TERM(GRAF04DC,GRAF05DC) ACTION(DENY,FAIL)
```

Time and Duration Restrictions

By default, an authorized resource can be used at any time. The TIME, TIMEREC, CALENDAR, DAYS, FOR, and UNTIL keywords can be used to place time and duration restrictions on this authorization.

For example, the following allows USER01 use of terminal GRAF04DC until midnight, December 31, 2002.

```
TSS ADDTO(USER01) TERM(GRAF04DC) UNTIL(12/31/02)
```

Continuing our example, the following allows USER01 use of the terminal until the indicated date, but now only between the hours of 9 A.M. and 5 P.M.

```
TSS PERMIT(USER01) TERM(GRAF04DC) UNTIL(12/31/02) TIME(09,17)
```

Note: The date field in the UNTIL keyword must follow the same mask as set by the DATE control option.

TZONE can be used as an automatic correction factor when there is a difference between the user's time zone and the CPU's time zone.

For example, if USER01, using a California terminal communicating with a New York CPU, needs to be restricted to using this local terminal during normal business hours, the TZONE keyword can simplify matters:

```
TSS ADDTO(USER01)  TZONE(-3)
TSS PERMIT(USER01)  TERMINAL(GRAF04DC) TIMES(08,17)
```

These two entries restrict USER01 to using local terminal GRAF04DC between 8 A.M. and 5 P.M. Pacific Coast Time.

Note: The TZONE keyword can be used only with a CREATE or ADDTO function (but not PERMIT).

The TIMEREC keyword can be used instead of the TIME keyword in order to specify access to a resource in multiple time ranges defined in 15-minute intervals within a 24-hour day.

The CALENDAR keyword can replace the DAYS keyword to control access to a resource by specifying which days of the week within a particular year access is allowed.

Restricting Types of Access

The types or levels of access to some pre-defined RDT resources (minidisks, CP commands, DCSSs, OS/DOS data sets, and user-defined resources) can be closely controlled with CA Top Secret. The mechanism for controlling these resources is the ACCESS keyword of TSS PERMIT. For example, the following authorizes USER01 to read--but only read--all minidisks with MAINT as their highest-level qualifier.

```
TSS PERMIT(USER01) VMMD(MAINT) ACCESS(READ)
```

Note that, because the default access level is READ, the following has the same effect. (In other words, the access level can be omitted if the default access level is acceptable.)

```
TSS PER(USER01) VMMD(MAINT)
```

To control access to resources dynamically added to CA Top Secret's RDT Record, use ACLST (the list of access levels that can be specified for the resource) and DEFACC (the default access level for the resource). For instance, the following tells CA Top Secret that the user-defined resource can be accessed at the WRITE, READ, or MULTI level.

```
TSS ADDTO(RDT) RESCLASS(user-defined resource)
   RESCODE(rescode) ACLST(WRITE,READ,MULTI)
```

When no default access is specified, the access level for the newly defined resource is NONE.

To assign a default access level (other than NONE) to a resource, use the DEFACC keyword. For example, the following makes READ the default access level for the user-defined resource.

```
TSS ADDTO(RDT) RESCLASS(resclass) RESCODE(rescode) DEFACC(READ)
```

Later, the administrator can use TSS REPLACE to change the default level.

The following table shows the access levels that apply to minidisks.

Level	Link access for minidisks
READ	R,RR
WRITE	W
UPDATE	R,W,RR,WR
MULTI	M
MREAD	R,RR,M,MR
MWRITE	W,M,MW
SREAD	SR

Level	Link access for minidisks
SWRITE	SW
SMULTI	SM
EREAD	ER
EWRITE	EW
ALL	ANY link
NONE	NO link

The following table shows the access levels that can be specified for DCSSs.

Level	Meaning for DCSSs
SHR	Shared copy of DCSS may be loaded
NOSHR	A non-shared copy of DCSS may be loaded
FIND	Entry point of DCSS may be located
PURGE	DCSS may be purged
ALL	ALL access
NONE	NO access

The following table shows the access levels that can be specified for OS/DOS data sets and DASD volumes.

Level	Meaning for Data Sets	Meaning for Volumes
READ	Data set can be read (open for input)	All OS/DOS data sets on the volume can be read
WRITE	Data set can be open for write	All OS/DOS data sets on the volume can be written to
UPDATE	Data set can be open for simultaneous read and write	All OS/DOS data sets on this volume can be updated
ALL	Data set can be accessed in any manner	All OS/DOS data sets can be accessed in any manner
NONE	Data set cannot be used in any manner (overrides any explicit authorization)	Volume and OS/DOS data sets cannot be accessed

Note: VSAM clusters are not individually protected at the CP start I/O level. However, the use of VSAM is protected. A user must be PERMITted to the VSAM file extents on a given volume. Example: To access the VSAM extents on VOLSER(DASD01), the ACID needs the following PERMIT in its Security Record:

```
TSS PERMIT(USER01) DSN(SYSVSAM.VDASD01)
```

The following table shows the access levels that can be specified for VMMACH.

Level	Meaning for Virtual Machine
LOGON	ACID may log onto virtual machine with the ACID= keyword in the CP LOGON command
AUTOLOG	ACID may autolog this virtual machine without a password
GRPLOGON	ACID may log on to the virtual machine with the GRPUSER= option of the CP LOGON command
SUROGATE	ACID may issue SUROGATE SET/RESET against the virtual machine
APPC	ACID can logon through an APPC connection.
ALL	ACID can use all of the above functions against the machine
NONE	ACID can use none of the above functions against the machine

Facility Restrictions

To restrict a user's access to a specific resource through the VM facility, use a TSS PER FAC entry. For example:

```
TSS PER(USER01) VMMD(ACCTSPAY.) FAC(VM)
```

restricts USER01 to linking minidisks owned by ACCTSPAY virtual machine through the VM facility only. Remember that USER01 must be allowed to have access to the VM facility through an entry such as this:

```
TSS ADD(USER01) FAC(VM)
```

The following table shows the access levels that can be specified for DIRECTORY.

Level	Meaning for CMS
READ	Files within SFS directory can be read
WRITE	Files within SFS directory can be written
UPDATE	Files within SFS directory can be read and written

SYSID Authorization

The SYSID keyword gives security administrators more control over the authorization of resources and facilities. When specifying a PERMIT or adding a facility to an ACID, the SYSID keyword allows you to specify the SYSID (which is actually the SMFID) that the authorization applies to.

SYSID can only be used with ADDTO, REMOVE, PERMIT, and REVOKE. You can enter up to five SYSIDs on a command.

The following example allows the user to use the TSO facility, but only on the system that has an SMFID of TSO1.

```
TSS ADD(useracid) FACILITY(TSO) SYSID(TSO1)
```

Similarly, if a PERMIT is done with SYSID, that PERMIT will match only if the access is done on that system.

Making Resources Globally Accessible

Certain types of CP commands and minidisks can be made available to all users. Access to these resources is best defined to CA Top Secret on a global basis through the TSS PERMIT(ALL) function. For example, the following makes the MAINT 0190 disk (usually the CMS system disk) available to all users for READ access.

```
TSS PERMIT(ALL) VMMD(MAINT.0190) ACCESS(READ)
```

Of course, this minidisk must already be owned by someone. Remember, also, that the PERMIT(ALL) function applies to both defined and undefined users.

All the standard access restrictions (such as ACCESS, FACILITY, TIME, etc.) are available for use with globally accessible resources.

Typical access restrictions to common resources: Some typical access restrictions are indicated in the following list of representative candidates for being designated as globally accessible resources:

- CMS system disks
- CMS help disks

Overriding Globally Accessible Authorizations

A specific user or profile authorization overrides the global authorization. For example the following:

```
TSS PERMIT(DOSCICS) VMMD(MAINT.0190) ACCESS(NONE)
```

Overrides:

```
PER(ALL) VMMD(MAINT.0190)
```

Special PERMIT Options Using ACTION Parameter

Through TSS PERMIT ACTION entries, special processing options can be selected. The chart below lists all the different ACTION options and their basic functions:

ACTION(ADMIN)

Gives authority to another administrator requesting the ability to PERMIT and REVOKE resources outside of his scope.

ACTION(FAIL)

If the permit is used to allow or disallow access, the authorization is processed as if the user is running in FAIL mode.

ACTION(DENY)

Causes a security violation to be generated when the user requests access to the specified resource. The mode that applies to the user is still honored, unless ACTION(FAIL) is used in conjunction with this option.

ACTION(AUDIT)

Audits all accesses to this resource regardless of the mode or logging options of the user.

ACTION(PASSWORD)

Requires the directory link password to the minidisk after CA Top Secret verification has authorized access to the minidisk.

ACTION(NOTIFY)

A message is issued to the operator on any access to the specified resource granted by this permission.

ACTION(NODSN)

All data set-level restrictions are bypassed by CA Top Secret and the minimum and maximum access levels to all data sets are controlled by the PERMITted VOLUME access.

ACTION(EXIT)

Passes control to the Installation exit for all accesses to the specified resource granted by this permission.

ACTION(VMPRIV)

Controls what access rights are authorized for ambiguous commands such as the CP command QUERY DASD.

Bypassing Resource Security Checking

ADDing the NORESCHK attribute to an ACID allows that user to bypass all security checks for all resources except OS/DOS data sets and alternate ACIDs. For example:

```
TSS ADDTO(USER10) NORESCHK
```

Since NORESCHK is a very powerful attribute, its use must be carefully restricted. Note that auditing is also bypassed for resources accessed via NORESCHK (other than data sets and volumes).

To rescind a user's resource security checking bypass privilege, use a TSS REMOVE function. For example:

```
TSS REMOVE(USER10) NORESCHK
```

The administrative authority required to ADD/REMOVE the NORESCHK attribute is MISC9(BYPASS).

Using Multiple Access Authorizations

For each resource access request, CA Top Secret conducts a security validation search to determine whether the access request is granted. Sometimes CA Top Secret encounters more than a single pertinent CA Top Secret entry during this search. For example, suppose USER01 requests READ access to the MAINT.0191 minidisk while control option AUTH(MERGE) is in effect. With this AUTH setting, both the user's primary and all attached profile Security Records are always searched. In addition, USER01 is attached to profiles PROFA and PROFB.

During its validation search, CA Top Secret encounters two pertinent entries:

```
TSS PERMIT(PROFA) VMMD(MAINT.) ACCESS(MULTI)

TSS PERMIT(PROFB) VMMD(MAINT.0191) ACCESS(READ)
```

Because the second PERMIT is a "better match" to the resource to which access is being requested, it governs whether the request is granted. Since it does provide authorization, the request is approved.

Multiple authorizations can be used to efficiently "tailor" elements in an overall security structure. Our example above might apply to a situation in which a large group of users (PROFA) are allowed to read a family of minidisks (MAINT.), while only a subset of this group (PROFB) are given the authority to update a particular member of this family (MAINT.0191). However, multiple authorizations may be employed carefully in order to avoid unexpected results.

To eliminate authorization ambiguities while still allowing users great flexibility in setting up their security restrictions, CA Top Secret uses a validation algorithm to determine whether a particular access request is granted. Refer to the "The Access Validation Algorithm" chapter in the General Concepts Guide for a description of how this algorithm operates.

CA Top Secret treats separate TSS PERMIT entries designating the same ACID and resource as discrete permissions rather than consolidating them into one combined permission.

To see the implications of these discrete permissions, consider these two sets of PERMIT commands:

Example A

```
TSS PERMIT(ALL) TERM(GRAF05DC) TIMES(08,17) DAYS(WEEKDAYS)
```

Example B

```
TSS PERMIT(ALL) TERM(GRAF05DC) TIMES(08,17)
TSS PERMIT(ALL) TERM(GRAF05DC) DAYS(WEEKDAYS)
```

In example A, the permissions are consolidated and follow AND logic: In other words, both conditions must be satisfied. To use terminal, you must log on Monday through Friday during business hours only.

In example B, the permissions are discrete and use OR logic, which means the logon attempt must be either during business hours or during the week.

Note that, in certain instances, the historical order in which multiple authorizations were made can determine which CA Top Secret error message is issued. For example, assume these two PERMITs have been entered in the order indicated:

```
TSS PERMIT(USER01) VMMD(MAINT.) ACCESS(READ) DAYS(MON)
TSS PERMIT(USER01) VMMD(MAINT.) ACCESS(UPDATE) DAYS(TUES)
```

Now, if USER01 tries to update a MAINT minidisk on a Monday, CA Top Secret denies access and issues an error message telling USER01 that the request is denied because of no UPDATE access. However, if the order of the authorizations had been reversed, then the request is denied but the error message says that USER01 is only authorized to UPDATE the MAINT minidisk on Tuesdays.

Revoking Multiple Authorizations

Multiple authorizations--that is, PERMITs that specify both the same ACID and resource identifier--can be revoked by using a single TSS REVOKE function.

For example, the following revokes both PERMITs in the previous example.

```
TSS REVOKE(USER01) VMMD(MAINT.)
```

Removing an Individual Permisson

To remove one specific permission, list the user's or profiles' XAUTH data and then issue the revoke exactly as it is listed. For example, the output of the following TSS LIST command:

```
XA DATASET = SYS1.PROCLIB
     ACCESS  = READ
     PRIVPGM = IEBCOPY
```

Would let you revoke the following authorization:

```
TSS REVOKE(user) DSN(SYS1.PROCLIB) ACCESS(READ) PRIVPGM(IEBCOPY)
```

Even if there were other permissions with different access levels or other criteria, only this permission would be revoked. However, if you code the REVOKE with only the resource and no other parameters, then all matches of the resource will be revoked.

Note: The FOR parameter should not be used on a REVOKE command.

Resource Class Translation

Use resource translation to align resource ownership, permission, and reporting with the business use of a given facility. Resource class translation allows you to translate:

- A source resource class into a target resource class during the resource security validation process
- A single source resource class into multiple target resource classes depending on the facility definitions

Important! Defining new resource classes to the RDT and translating security validations to that class *does not* imply that all resources within the original class are protected within the new class. Insure that all resources that require protection are defined as owned and permitted for any classes defined as targets of resource class translation. Consider assigning the default protection attribute to any user defined resource classes used as targets for translation.

Example: resource class translation

In this example, a system running with multiple CICS regions is active at the same time and each CICS region represented by a unique facility within the matrix. No matter which CICS initiates a transaction the OTRAN resource class is assigned to the security validation process. The definition of resource ownership and the reporting and tracking of resource access and violations occurs within a single resource class. Defining resource class translation criteria in each CICS facility the OTRAN class translates into multiple classes for security validation purposes.

```
TSS MODIFY FACILITY(CICS1=RXLTADD(OTRAN:TRAN1))

TSS MODIFY FACILITY(CICS2=RXLTADD(OTRAN:TRAN2))

TSS MODIFY FACILITY(CICS3=RXLTADD(OTRAN:TRAN3))
```

When a user enters a transaction in CICS1, the OTRAN resource class is translated to TRAN1, and the security validation process checks for ownership and permission of the resource name within the resource class of TRAN1. Violation and access audit reporting includes both the original resource class and the actual resource class within the report entry if a translation is performed.

Chapter 5: Securing Resources

This section contains the following topics:

Securing Resources

CA Top Secret protects a wide variety of computer resources, but to protect them, CA Top Secret must know about them. The types of resources (such as data sets, volumes, terminals and minidisks) that CA Top Secret protects are listed in the Resource Descriptor Table (RDT). Many resource types are already automatically defined to the RDT at installation, however, you can add additional resource types (including site-defined resources) by using the TSS ADDTO command. See the Command Functions Guide for a complete list of what is included in the RDT.

The following list represents some of the most common types of resources CA Top Secret can protect.

Facilities	Resources	Commands
VM	Data sets	TSO commands
BATCH	Programs	ROSCOE commands
TSO	Terminals	VM CP commands
CICS	Readers	CA Product coommands
CA-IDMS	CPUs	
STC	User resources	
ROSCOE	Operator commands	
IMS	DASD volumes	
Other	Tape volumes	
	VM resources	
	TSO resources	

Facilities	Resources	Commands
	DB2 resources	
	CICS resources	
	IMS resources	
	Advantage CA-IDMS resources	

Ownership vs. Authorization

Securing resources is a two step process. Once the resource type, or class, is defined in the Resource Descriptor Table (RDT), then the resources themselves must be:

- Owned by an individual ACID (we recommend using a DEPT ACID)
- Permitted to additional ACIDs (if necessary)

Resource classes are identified through the RDT. To secure a specific resource within that class, that resource must also be owned by a particular ACID. That is, CA Top Secret can protect data sets (in MVS) and minidisks (in VM) because the data set and minidisk resource classes are included in the RDT. To protect the ACCTNG.UPD data set or the ABC123 minidisk, that data set or minidisk must be owned. For example, the ACCTNG.UPD data set may be owned by the Accounting Department whose ACID is ACCDEPT. The ABC123 minidisk may be owned by John Smith, whose ACID is USER01.

Note: Data sets and minidisks, as resource classes, are protected by default in IMPL Mode and in FAIL Mode (which is the mode in which CA Top Secret is delivered). If you are operating in any other mode, you can use the DEFPROT attribute to invoke default protection. DEFPROT, which can also be used for other resource classes, is discussed in greater detail in the “Securing Resources” chapter.

Ownership of a resource automatically implies full access to that resource. For other ACIDs to have access to that resource, they must be authorized to do so. CA Top Secret allows a great deal of specificity in defining exactly what authorization restrictions you want placed on the use of a resource by a particular user or profile group.

You can restrict access according to the following:

- Time of day (for example, between 9 a.m. and 5 p.m.)
- Day of week (for example, weekdays only)
- A particular period of time (for example, until December 31st)
- Facility (for example an ACID may only access a particular resource through TSO)
- Path (meaning it must be accessed through a particular program which, in turn must be loaded from a particular library).

- A particular access level (such as UPDATE or READ).
- Note: Access level restrictions are not applicable to all resource classes.

See the “Securing Resources” chapter for more information about securing resources.

Certain types of resources, such as TSO commands and Advantage CA-Roscoe monitors, cannot be owned. Access to these resources can be restricted using the Limited Command Facility (LCF). LCF and unownable resources are discussed in greater detail in the “Securing Resources” chapter.

About Resources

When you have defined users to CA Top Secret and assigned system entry restrictions to them, the next step in building your security database is to secure your resources. For the purposes of CA Top Secret, securing resources involves:

- Defining them to CA Top Secret
- Designing access authorizations depending on who needs to access that resource (and, in some cases, how, when, and where they can access it)

Users are defined to CA Top Secret through their ACIDs. Resources are defined first through the Resource Descriptor Table (as part of a particular resource class) and then through individual ownership by a particular ACID. For example, terminal PD01 is defined first as a member of the resource class TERMINAL and then as an individual resource owned by ACID USER01. In the event a resource belongs to an unownable resource class, protection is extended through the Limited Command Facility (LCF).

Resource access authorizations are implemented through the TSS PERMIT command and can be customized through the TSS command keywords. Depending on the resource class, you can restrict a user’s access to that resource by day, time, facility, program, or level. Resource protection can also be extended on a default or global basis.

After a resource has been defined, and the appropriate authorizations issued, any future requests to access that resource are filtered through the Security Validation Algorithim. Access depends on PERMITs the ACID has, how explicit those PERMITs are, and where they are stored (in the user, profile or ALL Records).

Resource Descriptor Table

By default, the CA Top Secret Resource Descriptor Table (RDT) already contains a number of predefined resource classes. Each of those resource classes is identified by a unique keyword, and has associated with it certain attributes.

Using the TSS command, you can customize the RDT by:

- Modifying existing resource attributes
- Adding new resource classes

To view the contents of your RDT, issue a TSS LIST(RDT) command.

The table following lists a majority of the predefined protected resource classes and corresponding resource class keywords for MVS while the next table lists the additional resource classes for VM. Many of the resource classes cited in table following are also available in a VM environment; however, for a more complete list of what is available for your particular environment see your *Reference Summary*, *Implementation* or *Command Functions Guides*.

Those resource classes that are considered unownable by default are indicated by a bullet (n).

Note: Several predefined VSE resources have also been added to the RDT. These resources are used when a command is propagated (through CPF) from anCA Top Secret secured VM or MVS node to an CA Top Secret secured VSE node. For a list and description of these VSE resources, see the *Command Functions Guide*.

For MVS

The following table lists the predefined resource classes for MVS.

Type	Resource Class Keyword
APPC transaction programs and profiles	APPCTP
APPC side information files	APPCSI
APPC logical units	APPCPORT Data Set DSN
Tape or DASD volume	VOL
JES nodes or readers	TERM
TCAM, VTAM, or BTAM terminal	TERM
Program	PROGRAM
CICS destinations	DCT

Type	Resource Class Keyword
CICS files	FCT
CICS journal entries	JCT
CICS programs	PPT
CICS temporary storage areas	TST
IMS Application Group Name (AGN)	APPL
IMS Data Base Definition (DBD)	DBD
IMS Program Status Block (PSB)	PSB
IMS 4.1 commands	CIMS
CPU	CPU
Owned user-defined resource type	UR1, UR2
Field	FIELD
CA-Tape Scratch tape (CATAPE)	ABSTRACT
Linkage editor authorization (ACI)	ABSTRACT
Installation-defined system resource	ABSTRACT
CA-1 security bypass control (XTD98000)	ABSTRACT
Advantage CA-IDMS sub-scheme	SUBSCHEM
Advantage CA-IDMS area	AREA
Operator commands	OPCMD
SMS managed volumes, database tokens	IBMFAC
DB2 resources	IBMGROUP and any keyword starting with DB2.
SMS management class	MGMTCLAS
VTAM applications	VTAMAPPL
Automatic ACID propagation	PROPCNTL
CICS CEMT functions	SPI
SMS storage class	STORCLAS
TSO logon account	TSOACCT
TSO user attributes	TSOAUTH
TSO performance group	TSOPRFG
TSO logon procedure	TSOPROC

Type	Resource Class Keyword
SPF panel	PGM or CMD/XCMD
TSO commands	PGM or CMD/XCMD
Advantage CA-Roscoe commands (monitors)	PGM or CMD/XCMD
CICS transactions	OTRAN or TRAN/XTRAN
Advantage CA-IDMS transactions	OTRAN or TRAN/XTRAN
IMS transactions	OTRAN or TRAN/XTRAN
Unowned user-defined resource type	USERx (where x is any keyboard character)

For VM

The following table lists the predefined resource classes for VM.

Type	Resource Class Keyword
CP Commands	CPCMD
Virtual Machine Usage	VMMACH
Minidisks	VMMDISK
Discontiguous Saved Segments	DCSS
Diagnose Codes	DIAGNOSE
Dialed Virtual Machines	VMDIAL
DASD Volumes	VOLUME
RSCS Nodes	VMNODE
OS/DOS Data Sets	DSN
VM Readers	VMRDR
CPUs	CPU
Terminals	TERM
IUCV	IUCV
VMCF	VMCF
SFS Administrations commands	SFSCMD
SFS directory and file security	DIRECTRY

Type	Resource Class Keyword
Data space security	DSPACE
CA Top Secret VM Batch Programs	PROGRAM

Several CA Top Secret interfaces for products supported by the CA Common Infrastructure Services (CA-CIS) are defined in the RDT Record and are reserved for CA product use. For a complete list of these resources, see the *Command Functions Guide* and the *Security Administrator's Guide* for the respective CA products.

Modifying Existing RDT Entries

Depending on how your security environment is set up, you may decide you need to modify certain attributes for a particular resource class. To do this you would use the TSS REPLACE command with the RESCLASS and ATTRIBUTE keywords. The following is a sample command:

```
TSS REPLACE(RDT) RESCLASS(VMMDISK) ATTR(DEFPROT)
```

In this example:

- TSS REPLACE signifies that one attribute is replaced by new a one.
- RDT signifies that the Resource Descriptor Table is where the change will occur and where the new information is stored.
- RESCLASS is theCA Top Secret keyword.
- VMMDISK designates the actual resource class by keyword, in this case VM minidisks.
- ATTR is theCA Top Secret keyword.

- DEFPROT designates which attribute replaces the current attribute. In this case, the attribute is DEFPROT. For predefined resource classes you can only modify the following attributes:
  - EXIT calls the installation exit for the resource class.
  - NOEXIT deactivates the installation exit calls for that resource class.
  - DEFPROT protects this resource class by default
  - NODEFPROT deactivates default protection for this resource class.
  - GENERIC allows authorizations for a similar set of resources within this resource class to be grouped under a single prefix (for example, all data sets beginning with the ABC prefix).
  - MASK supports masking characters.
  - NOMASK does not support masking characters.
  - NONGENERIC causesCA Top Secret to treat each authorization for a specific resource within this resource class as a fully qualified name. (Generic prefixing and masking are discussed later in this chapter.)

The following attributes—frequently referred to as AUTH options—can also be modified for predefined resource classes. They affect the manner in whichCA Top Secret checks user, profile, and the ALL records for resource authorizations. They are discussed in the Security Validation Algorithm section in this chapter and in greater detail in the AUTH Control Option section in the *Control Options Guide*.

- MERGE uses AUTH(MERGE) for access checking of this resource class.
- NOMERGE deactivates AUTH(MERGE) for this resource class.
- ALLMERGE uses the AUTH(ALLMERGE) option for access checking of this resource class.
- NOALLMERGE deactivates AUTH(ALLMERGE) for this resource class.

For further information about these attributes as well as a complete list of modifiable attributes for installation defined resources, see your *Command Functions Guide*.

Adding New Resources to the RDT

CA Top Secret security can also be extended to protect resources that are not currently listed in the RDT—such as resource classes that are part of IBM or other vendor software packages and even installation-defined resource classes. Normally, these types of resources would not trigger operating system interfaces to CA Top Secret and therefore no security calls would be issued. To implement security for them, customize CA Top Secret as well as system exits and application code.

You can use the following methods to add resources to the RDT:

- TSS ADDTO command
- TSS REPLACE command (to modify certain predefined RDT entries that have been reserved for this purpose)

Using TSS ADDTO

If you decide to use the TSS ADDTO command you must supply two important pieces of information:

RESCLASS

The eight character resource class name. This is the name which will be honored by the TSS command, logging and the security interface

RESCODE

The hexadecimal resource code (ranging from 01-3F or 101 to 13F) used for trace and logging purposes.

For example, you could enter:

```
TSS ADD(RDT) RESCLASS($PAYUP) RESCODE(3F)
```

You also have the option of designating access levels, default access levels, and other access operands. For more details see the *Command Functions Guide*.

Using TSS REPLACE

Another way to add a new resource to the RDT is to modify one of the existing resource classes.

The following shows you how to use the TSS REPLACE command to modify an existing resource class in the RDT:

```
TSS REP(RDT) RESCLASS(ABSTRACT) ATTR(DEFPROT) ACC(READ)
```

The following resource classes have been reserved for any new resource:

ABSTRACT

Which, with additional site-written code, is used to extend security to other system type resources, such as job execution classes.

FIELD

Which is used to define database fields and to validate access to these fields by user program or transactions.

UR1, UR2

Which are used to designate owned, non-system, site-defined resources, such as particular account codes. They are implemented and function in the same manner as ABSTRACT and FIELD.

USERx

Which is used to designate unowned types of resources.

Any installation-defined resource class can be implemented as ownable or unownable. Your choice depends on how security is administered at your site.

Assigning Ownership

Once you know that the resource class is defined to the RDT, the next step is to determine which ACIDs should have ownership of the individual resources within that class (provided that resource class is ownable). You accomplish this through the TSS ADDTO command.

All of the ACID types discussed in the “Identifying Users” chapter can own a resource, however, how you distribute resource ownership throughout the ACID hierarchy is very important. The next sections discuss:

- Ownership by individual ACIDs versus ownership by Department, Division, or Zone
- Using Generic Prefixing and Minidisk/Dataset Masking to minimize the number of TSS commands you need to enter.

Selecting Resource Owners

Ownership by an individual ACID automatically implies full (ALL) access to that resource. For example, if USER01 was given ownership of the PAYROLL.UPDTE data set through the following command, then USER01 has ALL access to the PAYROLL.UPDTE data set:

```
TSS ADD(USER01) DSN(PAYROLL.UPDTE)
```

Ownership of a resource by a Department ACID, however, does **not** imply automatic access to that resource for all users in that Department. This is because a Department ACID (like Zone and Division ACIDs) represents a group of ACIDs and is not an actual user. A Department ACID does not sign on or function as an individual user does, therefore, it cannot access the resource. Resources are assigned to a Department ACID much the same way books are placed on a bookshelf—for safekeeping.

Owing to this structural difference, each user in that Department would have to be explicitly authorized to access that resource and this enables you to restrict access to an as-needed basis. For example, the following designates the Financial Department (FINDEPT) as the owner of the INVEST.RES data set:

```
TSS ADD(FINDEPT) DSN(INVEST.RES)
```

If USER02 belongs to that department he would still have to be authorized (through the TSS PERMIT command) to access the INVEST.RES data set. That way, you can give USER02 UPDATE access or even restricted to READ only access. The same applies for resource ownership by Divisions and Zones. For more information about TSS PERMIT and access level restrictions, see the How Do I Control Access to These Resources? section later in this chapter.

In general, you should assign ownership of your resources to the appropriate Department, Division, or Zone ACID. One exception to this guideline, however, might be to let users own all of their own minidisks and all the files whose data set name prefix matchs their userids.

Generic Prefixing

Assigning ownership for each individual resource may sound like an enormous task, but by using profile ACIDs and Generic Prefixing whenever possible you can greatly reduce the number of CA Top Secret command entries required.

You can define resources to CA Top Secret by their full name or through a generic prefix. Generic prefixing allows a group of similar resources in the same resource class to be defined to CA Top Secret simultaneously. For example, the IEHPROGM, IEHINIT, and IEHLIST programs could be grouped under the generic prefix “IEH.” That way, instead of entering:

```
TSS ADD(USER01) PGM(IEHPROGM)
TSS ADD(USER01) PGM(IEHINIT)
TSS ADD(USER01) PGM(IEHLIST)
```

you would enter:

```
TSS ADD(USER01) PGM(IEH)
```

Note: If you had added the NONGENERIC attribute to the PGM resource class in the RDT, you would enter:

```
TSS ADD(USER01) PGM(IEH(G))
```

That way the G would indicate that IEH is a generic prefix and not a fully qualified resource name.

Similarly, using generic prefixing, you could grant user ADM10MK access authorizations to all data sets whose first qualifier is ADM10.

Using Minidisk/Data Set Masking

Minidisks and data sets are the most frequently protected and the most numerous resources. CA Top Secret provides an additional means of minimizing TSS command entries: masking. Unlike generic prefixing, which can be employed with any type of resource, masking (or patterning) can only be used for certain types of resources. Minidisks and OS/DOS data sets, for example, are two resource types that support this concept. For information on whether you can use masking for a resource, see your *Command Functions Guide*.

Like generic prefixing, masking is used to group several resources with similar names; however, masking is **not** restricted to prefixes. Similar characters can occur in the beginning, middle, or end of the resource name and there can be any number of variable characters in between. Masking substitutes special characters for these variables. There are five different masking options:

Variable character substitution

Use the asterisk ***** to represent from zero to eight characters. For example, SAL*M.MARCH would match SALPAYM.MARCH, SALM.MARCH, and SAL.C.M.MARCH.

Index substitution

Use **.*.** or ***.** to instruct CA Top Secret to ignore an index level. For example, SAL.*.ABC would match SAL.BCKUP.ABC, SAL.ABC, and SAL.US.REC.ABC.

Fixed position

Uses the special character + to represent single positions within a resource specification. For example, SAL.+++.ABC would match SAL.REC.ABC but not SAL.BCKUP.ABC.

ACID substitution

Use the percent sign % to indicate that the ACID of the access requestor is substituted at run time. For example, using ACCT.%.ABC, USER01 could access ACCT.USER01.ABC but not ACCT.USER02.ABC.

Floating pattern

Use the hyphen - to represent a variable number of characters. For example, SAL-XMPT would match SAL.UPDTE.XMPT, and SAL.MSTR.XMPT.QTR.

Floating-pattern masking cannot be used in combination with other masking characters and it cannot be used for minidisks.

If you decide to use masking, make sure that you tell users which mask (or combination of masks) you want to use and how you want those masks to work.

Resource Access Control

Once you have identified your resources to CA Top Secret, the next step is to design access authorizations for each ACID that will need to use those resources. An ACID's authorization to a resource is determined by the PERMITs found in his user record, in the profile records attached to the user, and the ALL Record (which lists globally accessible resources). Access to ownable resources is restricted through the TSS PERMIT command and can be customized through the use of keywords. Unownable resources are restricted through the TSS ADDTO command and have limited customization capabilities.

This section introduces you to the following:

- The TSS PERMIT command
- Several TSS command keywords used for PERMIT customization
- Five special cases of resource protection, including
  - Limited Command Facility (LCF) for MVS
  - Owned Transactions (OTRAN) for MVS
  - Globally Accessible Resources
  - Default Protection through the DEFPROT attribute
  - Assigning bypass access checking attributes

  Note: OTRAN overrides LCF.

Using the TSS PERMIT Command

The TSS PERMIT command is used to designate which ownable resources an ACID is allowed to access. To enter the command you need to supply the ACID being granted access, along with the resource class and the name of the individual resource. For example:

```
TSS PER(USER01) TERM(PD01)
```

lets USER01 access the PD01 terminal. If you were using generic prefixing you could use the following command to permit USER02 to access all programs beginning with the IEH prefix:

```
TSS PER(USER02) PGM(IEH)
```

By adding the appropriate keywords to the TSS PERMIT command you can restrict that ACID's access by designating the following:

- Source of access (a designated terminal, CPU, and so on)
- Time/Date of access (9 to 5, weekdays, until 5-31-92, and so on)

  Note: Time restrictions are designated on a 24 hour clock; therefore, rather than specifying 9 to 5 you would specify 09,17.
- Facility access
- Path of access (through a particular program, loaded from a particular library)
- Access level (READ, WRITE, UPDATE, ALL, and so on).
- SMFID access
- ACTION taken byCA Top Secret when a particular access request is received

Note: These restriction options are not available for all resource classes.

Restricting Access by "Path"

By using the PRIVPGM keyword with the LIBrary and FACility keywords, you can design an ACID's access authorization so that that ACID would only be able to access a resource through a specific program that would have to be loaded from a specific library that, in turn, could only be accessed through a specific facility. For example:

```
TSS PER(USER02) DSN(PAYROLL.MASTER) PRIVPGM(PAYUPDAT)
   LIB('PAYROLL.PRODLIB') FAC(BATCH) ACCE(UPDATE)
```

requires USER02 to load the PAYUPDAT program from the PAYROLL.PRODLIB library, through BATCH to access the PAYROLL.MASTER data set.

Restricting Access by Levels

In the previous example, the ACCEss keyword was used to specify that USER02 has UPDATE access to the data set. The types of access levels to which you can restrict a user depend on the type of resource class involved, but some of the more common access levels are:

- READ
- WRITE
- UPDATE

- CREATE
- NOCREATE
- SCRATCH
- ALL
- NONE

For resources such as CICS and IMS databases,CA Top Secret recognizes access level keywords that correspond to the actual system terminology (such as BROWSE, REPLACE, PURGE, and FEOV).CA Top Secret also uses three access level determinations that you may not be familiar with. They are:

CONTROL

For a data set this allows VSAM control interval processing. For a volume, when used with CREATE, this allows the creation of data regardless of ownership.

FETCH

Enables access to a program library (load library) for the purposes of executing programs.

BLP

Lets a user access a tape bypassing the information kept in the label of that physical tape.

Restricting Access by SMFID

You can control access to resources and facilities by specifying which SMFID the authorization applies to. This type of control lets you specify from which system a resource or facility can be accessed.

Access is restricted through the ADDTO or PERMIT command functions and the SYSID parameter. You can specify up to five SYSIDs on a command. The following example lets the user use the TSO facility, but only on the system that has an SMFID of TSO1.

```
TSS PER(USER03) FACILITY(TSO) SYSID(TSO1)
```

Note: The PERMIT will match only if the access is done on the specified system.

Using the ACTION Keyword

Not all resource classes can be restricted by access level or path. For this reason, CA Top Secret provides another layer of security through the ACTION keyword. The operand specified with the ACTION keyword tells CA Top Secret how to respond to an access request. For example:

```
TSS PER(USER03) TERM(PD01) ACTION(NOTIFY)
```

tells CA Top Secret to notify the security console when USER03 accesses the PD01 terminal.

In addition to ACTION(NOTIFY), the ACTION operands include the following:

FAIL

Causes any unauthorized access attempt to be treated in FAIL mode regardless of the security mode the facility or user is in.

DENY

InstructsCA Top Secret to deny the ACID access to the resource.

AUDIT

Creates an audit trail when the resource is accessed.

EXIT

Invokes theCA Top Secret Installation Exit.

VMPRIV

Grants the accessor the privileged form of CP commands and DIAGNOSE instructions.

ADMIN

Allows a Security Administrator to administer resources that are not owned within his scope of authority. More information on resource administration is included in the "Administering Your Security Environment" chapter.

PASSWORD

Adds additional password protection before granting access to a data set or minidisk.

NODSN

Instructs CA Top Secret to only check volume authorizations for access requests to data sets on this particular volume.

Note: ACTION(DENY) does not apply to MVS data sets and volumes. Instead you can specify ACCESS(NONE).

Limited Command Facility

If a resource is considered unownable, CA Top Secret lets you extend security through the Limited Command Facility (LCF). LCF security is administered through the TSS ADDTO command and allows the Security administrator to limit the following:

- Commands available to a TSO user
- Monitor commands available to Advantage CA-Roscoe, ACEP, WYLBUR, and other online users

- Transactions an online user can use in CICS, IMS, and Advantage CA-IDMS
- Programs a batch job can execute

LCF protection can be employed following an inclusive approach—using the CMDS or TRANS keywords—or an exclusive approach—using the XCMDS or XTRANS keyword.

Note: The CMDS, TRANS, XCMDS, and XTRANS keywords are interchangeable.

The Inclusive Approach

The inclusive approach restricts an ACID to using only specifically authorized commands for a particular facility. For example:

```
TSS ADDTO(PROF01) TRANS(CICSPROD,(DDYA,DDYU,DDYY))
```

restricts the users connected to the PROF01 profile to using only the DDYA, DDYU and DDYY commands under CICSPROD.

The Exclusive Approach

The exclusive approach allows an ACID to use all but a specific list of restricted commands for a particular facility. For example:

```
TSS ADDTO(USER01) XCMDS(TSO,(SPF2))
```

allows USER01 to use all TSO commands except the SPF EDIT (SPF2) panel.

Through LCF, you can assign authorizations and restrictions to transactions on a user-by-user basis without impacting other users. However, because transactions controlled with LCF are not owned, if factors such as administrative scope and resource auditing capabilities are considered highly important for your security environment, an LCF-based approach to transaction security may not be the best approach for you.

Owned Transaction Security for MVS and VSE

The OTRAN resource class was introduced in CA Top Secret to allow transactions—which are ordinarily considered unowned—to be administered as owned resources. By allowing transactions to be owned by various zones, divisions, or departments, the following applies:

- Security can be administered on a decentralized, application-by-application basis, and
- Access to each transaction can be be tailored with the same flexible restriction options available for other resources—such as day, time, level, path, and so on.

If a transaction is owned, it is protected from use by any user in any facility.

Consider all the users who may need that transaction (in both Test and Production regions) before using the OTRAN resource class.

Note: Transaction security through the OTRAN resource class overrides Limited Command Facility (LCF) protection. Once a transaction is owned, any appearance of it in an LCF list, whether inclusive or exclusive, is ignored.

Globally Accessible Resources

You may want to make a particular resource—such as system and compiler libraries, or storage and working DASD volumes—globally accessible to all users. To do so, simply add that resource to the ALL Record. Although the resource is being added, you need to use the TSS PERMIT command. For example:

```
TSS PERMIT(ALL) TERM(PD01,PD02)
```

allows all users—both undefined and defined—to access terminals with the PD01 and PD02 prefixes. When you add a resource to the ALL Record, you can still specify other access restrictions, such as TIME and FACILITY, that would normally be available for that resource.

Note: A specific access authorization in a User or Profile Record overrides a global authorization in the ALL Record.

DEFPROT Attribute

In FAIL mode, volumes, data sets and minidisks are automatically protected by default. In WARN and IMPLEMENT modes, you add default protection by modifying the RDT (using TSS REPLACE instead of TSS PERMIT) to include DEFPROT for that resource class. For example, if you were in WARN mode but you wanted to protect all data sets by default, you would enter:

```
TSS REP(RDT) RESCLASS(DSN) ATTR(DEFPROT)
```

In WARN mode, access violations would be generated for those volumes, data sets, and minidisks that have not yet been defined toCA Top Secret.

Default protection is not limited to data sets and minidisks. Assigning the DEFPROT attribute to any resource in the RDT can give default protection to predefined—as well as dynamically defined—resources that are normally not protected by default in any mode, including FAIL.

Bypass Resource Checking

Earlier in this chapter you learned that by using ACTION(NODSN), along with the TSS PERMIT command, you can specify that no data set checking will be performed for that particular volume. In some rare cases, however, you may want to extend that bypass privilege so that any time a particular ACID makes a certain type of access request, those requests will bypass security checks.

Note: We do not recommend this type of blank check security access for ordinary usage. Limit its use to setting up disaster recovery procedures.

You can do this by simply adding one or more bypass attributes to the ACID. This type of action is not recommended for most users. The following list of bypass options (also called no-check attributes) should be used with great care if at all. They are as follows:

NORESCHK

Allows the ACID to bypass all resource checking with the exception of data sets and volumes.

NOVOLCHK

Allows the ACID to bypass all volume checking.

Note: If data set access it being requested,CA Top Secret responds according to the data set authorizations. To allow an ACID unrestricted access to an entire volume, you must also add the NODSNCHK attribute.

NODSNCHK

Allows an ACID to bypass data set checking. Not to be confused with ACTION(NODSN) which only affects access requests for a particular volume.

NOLCFCHK

Allows an ACID to bypass LCF checking.

NOSUBCHK

Allows an ACID to bypass job submission security checking. This way an associated ACID can submit all jobs regardless of the (derived) ACID on the job card being submitted.

NOVMDCHK

Allows an ACID to bypass VM minidisk link checking.

Add these attributes to an ACID through the TSS ADD command, and rescind them through the TSS REMOVE command. For example, the following allows the SUPRACID to bypass all resource checking:

```
TSS ADD(SUPRACID) NORESCHK NOVOLCHK NODSNCHK NOVMDCHK
```

While the following example removes those privileges:

```
TSS REM(SUPRACID) NORESCHK NOVOLCHK NODSNCHK NOVMDCHK
```

To administer these no-check attributes, the Security Administrator must have the appropriate administrative authority.

VM Protection Examples

CA Top Secret provides the ability to protect resources that are specific to VM. This is accomplished through the use of VM keywords.

Some of the most commonly protected VM resources include:

- CP commands
- Minidisks
- VM readers
- DCSSs
- RSCS nodes
- Diagnose codes
- VM DIAL commands
- Program Protection
- IUCV
- VMCF

Control Program (CP) Commands Protection

In CA Top Secret, Control Program (CP) commands are grouped into three categories:

- **General user commands**—A general user command is one that only requires a class G user status in a standard VM operating system environment, and one that means only one thing to all users. For example, the SPOOL command would perform the same function regardless of whether is was issued by a user with a class B or a class G status. Other examples of general user commands include LINK, TAG, SEND.

  By PERMITting ACIDs to use these general commands, the CA Top Secret administrator gives access rights similar to rights accorded class G VM users.

Privileged commands—Like general user commands, privileged commands also perform the same function regardless of the user's class status. Unlike general user commands, only a few select users are given the class that allows them to use these privileged commands. Examples of privileged commands include ATTACH, SHUTDOWN, NETWORK, STORE HOST

In a native VM environment, these commands require a specific class, other than class G. Such commands are considered by CA Top Secret to have implied VMPRIVilege access rights. If USER01 is PERMITted to the CP command SHUTDOWN, CA Top Secret makes USER01 a privileged VM user of that command only—whether VM allows it or not.

- **Mixed commands**—Unlike the previous command types, these commands **do** perform different functions depending on the class of the user who issues them. For example, a class B user who types in QUERY DASD receives information about the **real** DASD associated with the **real** CPU while a class G user who types in the same QUERY DASD command receives information about the **virtual** DASD associated with his **virtual** machines. Another example of mixed commands is SET.

Note: In FAIL mode,CA Top Secret authorizations supersede VM's privilege classes.

Minidisk Protection

From a VM system user's point of view, minidisks are shared DASD devices of various sizes, which are accessed in one of two ways:

- Through a VM directory MDISK (minidisk) or LINK statement
- With the CP command LINK

Like anyone else, CA Top Secret users get access to their own minidisks at logon but with CA Top Secret, you can restrict users' access to one another's minidisks. For example, the following gives the DCA for the Finance Department (FINDCA) READ access for all of USER01's, USER02's, and USER03's 019x minidisks ranging from 0190 through 019F:

```
TSS PERMIT(FINDCA) VMMDISK(USER01.019,USER02.019,USER03.019)
   ACCESS(READ)
```

As far as VM is concerned, users own minidisks because of MDISK statements in their directories. When users LINK to their own minidisks, no security is needed (not even CA Top Secret). When one user tries to LINK to another user's minidisk, however, CA Top Secret provides security checking.

VM Reader Protection

Each virtual machine may have one or more virtual readers. Readers are the common mechanism for submitting work from one virtual machine to another on the same real CPU. CA Top Secret protects virtual readers by restricting, through the VMRDR keyword, who is allowed to place files in a virtual reader. For example, the following calls CA Top Secret security and controls any spooling activity directed toward the CMS batch reader:

```
TSS ADDTO(USER01) VMRDR(CMSBATCH)
```

Reader protection also extends to other readers, such as RSCS readers and readers for other operating systems (such as MVS) working under VM.

CA Top Secret also provides extended security controls for the CP SPOOLing commands—TRANSFER, CHANGE, ORDER, and PURGE—through the CPCMD keyword. For example, the following permits USER01 to issue a TRANSFER against his own spool files (with what amounts to a class G access right):

```
TSS PERMIT(USER01) CPCMD(TRANSFER)
```

Conversely the following lets USER01 transfer any spool files from any user (normally a CP class D privilege):

```
TSS PERMIT(USER01) CPCMD(TRANSFER) ACTION(VMPRIV)
```

DCSS Protection

Most often, discontiguous saved segments (DCSSs) are used to share READ only storage between two or more virtual machines. An example is a DCSS for CMS. The CMS DCSS is more than 300K bytes. By sharing a DCSS, all CMS machines can point to one 300K+ section of memory, instead of having one copy per machine. CA Top Secret protects both shared and unshared discontiguous saved segments (DCSS). With the DCSS keyword and the TSS PERMIT command, you can secure standard program products such as PROFS, SAS, and VSAM that run as a DCSS. For example, the following allows USER03 to use the shared portion of the SAS system which resides in a DCSS:

```
TSS PERMIT(USER03) DCSS(SAS)
```

RSCS Node Protection

When dealing with networks of virtual machines, the typical site uses RSCS to transfer files between virtual machines on separate real machines. RSCS also enables a VM system to become a node in a multi-system NJE network so that VM users can route jobs and output directly through RSCS to remote systems and printers. The target node is identified with the CP TAG command. The CA Top Secret keyword VMNODE is used with the TSS PERMIT command to specify what nodes a particular user can ship spool files to.

```
TSS PERMIT(USER01) VMNODE(DALLAS,CHICAGO)
```

authorizes USER01 to ship reports to the DALLAS and CHICAGO RSCS nodes.

Note: The CA Top Secret administrator who issues the PERMIT shown in the last example must have ADMIN authority for the DALLAS and CHICAGO nodes.

Diagnose Code Protection

The DIAGNOSE keyword lets you protect VM diagnose codes. For example:

```
TSS PERMIT(USER01) DIAG(84)
```

allows USER01 access to the 84 diagnose code. When listing the information for that ACID, you would see the code in its full 4-character format: 0084.

Dial Protection

CA Top Secret performs a security check for any VM user ID that has been defined to it. Any user who tries to DIAL a defined ID will find his request cleared by CA Top Secret. DIAL access is secured by using the VMDIAL keyword along with the TSS PERMIT command. For example:

```
TSS PERMIT(USER02) VMDIAL(USERA,USERB,USERC)
```

allows USER02 to dial to users USERA, USERB and USERC. Once a VMDIAL resource is owned, any ACID dialing those machines will be prompted for ACID and password. Furthermore, the ACID he provides must already be PERMITted to dial the given virtual machine.

Note: The ACID does not have to be defined in the VM directory, only in the Security File. CA Top Secret can provide security for users who do not have a VM user id.

You can also specify to which line in the virtual machine the ACID may dial. This is done by appending the line number to the end of the VMDIAL operand. For example:

```
TSS PERMIT(USER02) VMDIAL(USERA.002(G))
```

forces USER02 to dial into USERA's virtual machine GRAF devices 0020 through 002F.

Note: The Security Validation Algorithm treats VMDIAL resources differently depending on whether or not the user includes a specific line on the dial command. If the user does specify a line ('D USERA 0020'), Top Secret interprets the resource requested to be VMDIAL (USER 0020). To be allowed access to this resource, a user should be permitted to it explicitly or by a generic permit, as in the following examples:

```
TSS PERMIT(USER02) VMDIAL(USERA.0020)
TSS PERMIT(USER02) VMDIAL(USERA.002(G))
TSS PERMIT(USER02) VMDIAL(USERA(G))
```

Because VMDIAL is a NONGENERIC resource, a permit to VMDIAL(USERA) will not allow access to VMDIAL(USERA.0020). By contrast, if the user does not include a line number in the dial command ('D USERA'), then the requested resource is considered to be VMDIAL(USERA), and a permit to VMDIAL(USERA) would grant access to this resource. Because of the different ways in which a user can issue a dial command, administrators should exercise great care in defining permissions to a VMDIAL resource.

Program Protection

When submitting a batch job to the CA Top Secret server machine, an additional security check will be performed at the program level. Prior to r1.3, any administrator with the BATCH facility could submit and execute ANY CA Top Secret VM utility program. Now, in addition, the submitter must also be given access to the utility through the PROGRAM resource class. This gives the installation the ability to give, for example, all administrators use of the TSSUTIL utility, but only selected individuals the ability to run TSSRECVR.

IUCV and VMCF Protection

The Inter-User Communications Vehicle (IUCV) is a VM Control Program (CP) facility that permits a program running in a virtual machine to communicate with other virtual machines, with a CP system service, and with itself. The IUCV communication occurs between a source communicator and a target communicator and is used to:

- Create and dismantle paths
- Send and reply to messages

- Receive or reject messages
- Control the sequence of IUCV events.

CA Top Secret protects IUCV access to virtual machines during IUCV path creation through the IUCV keyword. For example, the following allows USER02 to establish an IUCV path with DVM01:

```
TSS PERMIT(USER02) IUCV(DVM01)
```

The Virtual Machine Communication Facility (VMCF) is another communications method provided by the CP component of VM. VMCF provides virtual machines with the ability to send data to and receive data from any other virtual machine.

CA Top Secret protects VMCF access to virtual machines during each VMCF SEND or SENDX operation through the VMCF keyword. For example, the following allows USER02 to initiate a VMCF SEND or SENDX operation to DVM01:

```
TSS PERMIT(USER02) VMCF(DVM01)
```

With Genlevel 9505, the following APARs make security checking for IUCV CONNECT optional (GO50617, GO50618, GO50619, GO50620, GO50616, and GO50621).

If you want to enable IUCV CONNECT security, you should apply one of the following special PTFs. You should also ensure that their secondary user is not connected to *MSG, or connected to any user connected to *MSG.

- GS50663 (370)
- GS50664 (VM/XA)
- GS50665 (VM/ESA 1.1.1)
- GS50666 (VM/ESA 1.2.0)
- GS50662 (VM/ESA 1.2.1)
- GS50667 (VM/ESA 1.2.2)

Making IUCV CONNECT optional was done to avoid possible system hangs if the server is processing any IUCV CONNECT requests for its secondary user, or for any ACID connected to the secondary user through *MSG.

Using an Updated Security Record

When administration is performed on an ACID, the actual changes are made to the security record on the Security File. Because the security record is brought into storage at logon time, the user must log off (or disconnect, in VM) and log back on to get the newer version of his security record.

The command TSS REFRESH automatically causes a new security record to be read from the Security File for the administrator who issues the command. An administrator can perform this function for another userid by adding the ACID to the command. For example, if DCA1 issues the following command:

```
TSS REFRESH USER1
```

USER1 will get another copy of their security record.

TSS REFRESH only applies to CA Top Secret.

Chapter 6: Administering Your Security Environment

This section contains the following topics:

DATA Authorities

DATA authorities designate which portions of a Security Record the Security Administrator can display when he issues a TSS LIST command. Some of the operands used with the DATA keyword include:

BASIC

Lists general ACID information, such as name, type and facility restrictions

SESSKEY

Lists the session key used to verify that one LU is authorized to link to another LU for the purposes of APPC conversation processing

WORKATTR

Lists the SYSOUT delivery and accounting information associated with the ACID

XAUTH

Lists resources PERMITted by an ACID within his scope, including information about what access levels are allowed and which ACID owns the resource

RESOURCE

Lists information about resources owned by an ACID

SOURCE

Lists information about the ACID's input device restrictions (such as terminals or CPUs)

ADMIN

Lists information about the ACID's administrative authorities

PASSWORD

Lists the ACID's password expiration date and interval

PWVIEW

Lists the ACID's actual password as well as the expiration date and interval

PROFILE

Lists the profiles attached to the ACID

EXPIRE

Lists the expiration date for a temporary profile or group.

In the case of the DATA keyword, the ALL operand does not confer, by itself, all of the associated authorities. You must also specify PASSWORD, PWVIEW, EXPIRE and SESSKEY (due to the sensitive nature of passwords and SESSKEY information) and PROFILE (because the profiles connected to an ACID may not fall under the scope of the Security Administrator). For example, to give the FINVCA security administrator the authority to list every possible item of Security Record information, you would have to issue the following command:

```
TSS ADMIN(FINVCA) DATA(ALL,PWVIEW,PROFILE,SESSKEY)
```

Note: The PWVIEW authority level is only honored if the PWVIEW control option is set to YES.

SESSKEY only applies to the APPCLU Record.

RESOURCE Authorities

RESOURCE authorities are used to designate what type of maintenance a Security Administrator can perform on resources. There are six operands that can be specified with the RESOURCE keyword:

OWN

Allows the designated administrator to define, move, and remove resource ownership

XAUTH

Allows the designated administrator to PERMIT or REVOKE resource access. The access levels must also be specified. For example:

```
TSS ADM(FINVCA) RES(XAUTH) ACCE(READ,FETCH)
```

only allows FINVCA to PERMIT an access level of READ or FETCH for the ACIDs within his scope

INFO

Allows the designated administrator to issue TSS WHOHAS and TSS WHOOWNS for resources

AUDIT

Allows the designated administrator to ADD or REMOVE resources from the AUDIT record

REPORT

Allows the designated administrator to use BATCH reporting utilities for resources (see the “Monitoring Your Security Environment” chapter for a discussion of report running)

ALL

Allows the designated administrator to perform all of the above functions

The keyword, RESOURCE, applies to all resources. If you only wish to indicate a specific resource, substitute that resource class name. For example, the following gives FINVCA the authority to perform WHOOWNS and WHOHAS inquiries for any type of resource:

```
TSS ADMIN(FINVCA) RESOURCE(INFO)
```

The following example gives RESVCA the same type of authority-but only for the program resource class.

```
TSS ADMIN(RESVCA) PGM(INFO)
```

Note: In addition to specifying RESOURCE, you must also specify a particular ACCESS level, if that applies, otherwise the default is READ. For example:

```
TSS ADMIN(RESVCA) DSN(OWN) ACCE(CREATE,UPDATE)
```

FACILITY Authorities

FACILITY authorities govern which facilities the Security Administrator has authority over. The FACILITY keyword is used in conjunction with the other administrative authorities. For example, the following gives the FINVCA Security Administrator the authority to create and maintain ACIDs for CICS users and profiles:

```
TSS ADMIN(FINVCA) ACID(CREATE,MAINTAIN)
   FACILITY(CICSTEST,CICSPROD)
```

Note: CA Top Secret does not check an Adminstrator's facility authorization when he issues a TSS PERMIT allowing an ACID to access a resource only from a particular facility.

MISC1 Authorities

There are currently 5 “degrees” of miscellaneous authorities: MISC1, MISC2, MISC3, MISC8, and MISC9. MISC1 authorities generally pertain to ACID access restrictions, such as LTIME, suspensions and RDT maintenance. The following keywords are used to indicate what type of MISC1 authority a Security administrator has:

LCF(MVS only)

Allows the designated Security Administrator to assign LCF (Limited Command Facility) command and transaction restrictions. See the "Securing Resources" chapter for more information about LCF.

LTIME

Allows the designated Security Administrator to maintain automatic terminal locktime intervals through the LTIME keyword. For example, the following allows the FINVCA to administer the following command for USER01 (provided USER01 is in FINVCA's scope of authority).

TSS ADMIN(FINVCA) MISC1(LTIME)

For example:

```
TSS ADD(USER01) LTIME(30)
```

This would cause USER01's terminal to automatically lock after a 30 minute interval of inactivity.

SUSPEND

Allows the designated Security Administrator to suspend and unsuspend ACIDs by using the SUSPEND keyword with the TSS ADD/REMOVE commands.

TSSSIM

(MVS and VSE only) Allows the designated Security Administrator to use the TSSSIM utility

NOATS

(MVS and VSE only) Allows the designated Security Administrator to prevent ACIDs (within his scope) from signing on through Automatic Terminal Signon (ATS). This applies to the CICS, IMS and CA-IDMS facilities.

RDT

Allows the designated Security Administrator to maintain and list the RDT Record.

ALL

Allows the designated Security Administrator to perform all MISC1 type functions.

MISC2 Authorities

The MISC2 authorities basically pertain to special facility oriented resource administration such as:

TSO

Allows the designated Security Administrator to assign TSO-related data fields to an ACID

SMS

Allows the designated Security Administrator to assign SMS-related data fields to an ACID

PC

Allows the designated Security Administrator to assign PC group attributes

APPCLU

Allows the designated Security Administrator to maintain the APPCLU Record and assign APPC-related fields to an ACID

WORKATTR

Allows the designated Security Administrator to assign accounting and sysout delivery information fields to an ACID

TARGET

Allows the designated Security Administrator to assign nodes associated with ACIDs

MISC3 Authorities

SDT is the only authority level that can be used with MISC3. MISC3(SDT) authority allows the CA Top Secret security administrator to maintain and list the Static Data Table (SDT).

MISC8 Authorities

The MISC8 authorities basically pertain to the authority to list the RDT, FDT, STC, SDT, and use the SUSPEND keyword to remove administrative suspensions using the following operands:

LISTRDT

Allows the designated Security Administrator to list the Resource Descriptor Table and the Field Descriptor Table.

LISTSTC

Allows the designated Security Administrator to list the Started Task.

LISTSDT

Allows the designated Security Administrator to list the Static Data Table.

REMASUP

Allows the designated Security Administrator to issue the TSS REMOVE(acid) ASUSPEND command function that removes administrative suspensions.

MISC9 Authorities

The final category of miscellaneous authorities, MISC9, deals with higher level administrative authority and employs the following operands:

BYPASS

Allows the designated Security Administrator to assign any of the various security-bypass attributes. These are also called no-check attributes and are discussed in the “Securing Resources” chapter.

CONSOLE

Allows the designated Security Administrator to assign the CONSOLE attribute to any ACID within his scope

GENERIC

Allows the designated Security Administrator to issue generic WHOOWNS inquiries to obtain a list of all resources owned within his scope

TRACE

Allows the designated administrator to ADD/REMOVE the TRACE attribute

MODE

Allows the designated administrator to specify security modes for users

GLOBAL

Allows the designated administrator to PERMIT/REVOKE access to the ALL Record

ALL

Allows the designated administrator to perform all of the MISC9 type functions

SCOPE Authority

SCOPE authority is used by the MSCA to determine the administrative scope of an LSCA (the only control ACID type with variable scope).

That scope may include Zones, ZCAs, and even other LSCAs. For example:

```
TSS ADMIN(LSCA01) SCOPE(LSCA02,LSCA03,ZCA01)
```

designates LSCA02, LSCA03, and ZCA01 as being within the administrative scope of LSCA01.

SCOPE authority does not apply to any other control ACID and may not be defined by any control ACID other than the MSCA.

Modifying Your Security Environment

TSS commands are used to build your security database. Control options are used to define your security environment. How your security environment is designed can make a major difference in how CA Top Secret responds to both resource access requests and security violations. Through control options a Security Administrator can tell CA Top Secret:

- How to process normally as well as under specific MODEs and circumstances
- What password selection rules and violation thresholds are in effect
- Which activities should be logged
- What actions should be taken after shutdown or maintenance
- Which features, facilities, and products are on the operating system and how they should be secured (see the “Administering Facility Security” chapter)

A complete discussion on control options and managing your security environment can be found in your *Control Options* and *Implementation Guide*, but for the sake of brevity, this chapter will concentrate on only the following control options:

- MODE (for modifying your CA Top Secret security mode)
- NEWPW (for designing password specifications)
- LOG (for indicating what security events to log, and how)

- VTHRESH (for setting the violation threshold)
- PTHRESH (for preventing “password guessing”)

Using Control Options

At installation, your CA Top Secret security environment is automatically defined by the control option default values set in the Parameter File.

As part of the installation process, you are asked to view the contents of this file to verify that the default values listed are the values you want. At this point, changes made to control options in the Parameter File are automatically in effect the moment CA Top Secret is activated.

To view the contents of this file, you would simply edit/browse the sample Parameter File distributed with CA Top Secret.

In MVS, issuing a TSS MODIFY(STATUS) command would present you with a list of control options similar to the list on the following page. For a VM example, see the “Monitoring Your Security Environment” chapter.

```
TSS MODIFY(STATUS)
TSS9501I SECURITY FILE(014%)  ID=PRIMARY   SHRFILE(NO)      ENQSIZE(03072)
TSS9502I LAST CHANGED ON 03/21/95 AT 08:15  BACKUP(ACTIVE-15:00)
TSS9503I AUDIT FILE(006%)  RECOVERY FILE(003%)
TSS9504I TIMER(015)  SYSOUT(A,LOCAL)  CANCEL(NO)  SWAP(NO)
TSS9505I AUTH(OVERRIDE,ALLOVER)  MODE(WARN)  DOWN(BB,SB,TB,OW)
TSS9505I FACMODE(B W,S W,T W,I F,J W,5 W,A W,6 W,7 W,4 W,3 F,2 F,1 W,APW,HSW,CNW)
TSS9506I LOG(ACC, INIT,SEC9,MSG)  LOGBUF(0032)  CURRBUF(0013)
TSS9507I JOBACID(U,7,0)  SUBACID(U,7)
TSS9508I JES(SSID=JES2,TYPE=JES2,LEVEL=SP 4.3.0,VERIFY)
TSS9509I JCT(INDEV=0320,ROUTE=0316,NJHDR=0000)
TSS9534I NJEUSR(*NONE*)  JESNODE(A51SJES2)
TSS9510I TEMPDS(NO)  ADSP(NO)  AUTOERASE(YES)  TAPE(OFF)
TSS9511I PWEXP(030)  PWHIST(03)  HPBPW(002)  PWVIEW(YES)
TSS9512I NEWPW(MIN=4,WARN=03,MINDAYS=01,ID,TS,RS)
TSS9513I PTHRESH(003)  VTHRESH(002,NOT)  INACTIVE(030)
TSS9514I NPWRTHRESH(2)  MSUSPEND(YES)
TSS9515I EXIT(OFF)  DATE(MM/DD/YY)
TSS9516I CPF(ON)  CPFWAIT(NO)  CPFTARGET(LOCAL)
TSS9533I CPFRCVUND(NO)
TSS9517I CPFNODE(XE14)    STATUS(ACTIVE,SPOOL,PRE50)
TSS9517I CPFNODE(XE38)    STATUS(ACTIVE,SPOOL,PRE50)
TSS9517I CPFNODE(XA25)    STATUS(ACTIVE,SPOOL,PRE50)
TSS9517I CPFNODE(XA10)    STATUS(ACTIVE,SPOOL,PRE50)
TSS9517I CPFNODE(XE05)    STATUS(INACTIVE,SPOOL,RETRY,SEND-ONLY,PRE50)
TSS9518I PRODUCTS(TSO/E)
TSS9519I ADABAS(N/A)  DL1B(NO)
TSS9520I DUFPGM(*NONE*)
TSS9521I DB2FAC(DSNK=DB2PROD,DSNL=DB2PROD)
TSS9522I TEXTTSS(QA MACHINE MVSXE51)
TSS9525I PCADMIN(SYSADM)
TSS9526I PCIDLE(00)     PCDSDAYS(*NULL*)
TSS9527I PCMINPWD(1)    PCLGTYPE(1)
TSS9528I PCOPTS(NOCONNECT,SGNOFFMSG)
TSS9529I MFACCESS(DORM)
TSS1304I ------ CACHE STATISTICS -----
TSS1305I MAXSIZE ( 000002000K ) SIZE ( 000000914K )
TSS1306I CALLS ( 000010259 ) SATISFIED ( 000008144 )
TSS1307I CLEARED ( 000000000 )
TSS0300I MODIFY  FUNCTION SUCCESSFUL
```

In addition to changing control option values in the Parameter File, you also have the option to temporarily override those values by using any of the following four methods. However, since changes made through these methods are temporary, they will not be reflected in the event CA Top Secret is brought down and restarted.

- PARM File (MVS only)-When the CA Top Secret started task procedure is executed this method can be used to override control option default or Parameter File specifications. The PARM field is limited to 100 characters,
- O/S console with O/S start TSS Command (MVS only)-This method can be used when the CA Top Secret started task procedure is started up, and overrides changes made through the Parameter File or the PARM field.
- O/S console with O/S modify TSS command (MVS only)- This method can be used at any time while CA Top Secret is running and it overrides changes made through any of the previous three options.
- TSS Modify Command (VSE, MVS, and VM)-In addition to displaying the contents of the Parameter File, TSS MODIFY can also be used, at any time, to override the values specified by any of the previous methods. TSS MODIFY is used the the same way as any other TSS command. For example:

  ```
  TSS MODIFY(FAC(TSO=MODE=IMPL))
  ```

  tells CA Top Secret to modify the FACILITY control option (discussed later in this chapter and in the "Administering Facility Security" chapter) so that users on the TSO Facility will be in IMPL MODE (also discussed later in this chapter).

Determining Your CA Top Secret Security Mode

The MODE control option determines how CA Top Secret responds to security violations. There are four settings for the MODE control option:

DORMant

CA Top Secret is installed, but it is not actively validating activity. The TSS command is fully supported and can be used.

WARN

CA Top Secret is active, but violations do not result in requests being failed. This mode gives the Security Administrators the opportunity to fine-tune definitions without impacting processing activity. They can examine security incident reports (see the “Monitoring Your Security Environment” chapter for a discussion of reporting and logging options) and adjust authorizations accordingly. Depending on how the MSG control option is set, violation messages may or may not be sent to the user.

IMPLement

CA Top Secret is active and will fail any unauthorized access requests. Users not defined to CA Top Secret can operate normally but will be restricted from accessing protected resources.

FAIL

CA Top Secret is in full control of access requests. All users must be defined and all resources protected. Unauthorized access requests will be failed.

In a typical security environment, the entire installation will gradually move from DORMANT to FAIL mode. This is called phased implementation, and it lets you both introduce your users to how CA Top Secret will impact their daily job performance, and to customize your security database and security environment as the need arises.

In addition to phased implementation-where the entire installation gradually moves from one mode to another-you can also specify concurrent security modes-where different parts of your installation are running under different security modes all at the same time.

Using Concurrent Security Modes

Security modes can be assigned implicitly-through the MODE control option and FACILITY suboptions-or explicitly-through the TSS PERMIT command.

When the security mode is set through a control option, that mode applies to the entire installation or, in the case of the FACILITY/MODE suboption (discussed in the next section) to a specific facility (such as TSO or VM).

For example, the following uses the MODE control option to place the entire installation in WARN mode.

```
MODE(WARN)
```

While the following example tells CA Top Secret to treat the VM production facility as though it was in IMPL mode.

```
FAC(VMPROD=MODE=IMPL)
```

When the TSS PERMIT command is used, the Security Administrator can assign a specific security mode to a particular user or profile ACID. For example, the following places USER01 in WARN mode, regardless of what mode the rest of your site is in.

```
TSS PER(USER01) MODE(WARN)
```

Determining Password Restrictions

ACID passwords are assigned using the TSS CREATE command. However, the guidelines that must be followed when users change their passwords and the way in which CA Top Secret responds to password changes is determined by the NEWPW control option and its various suboptions.

The following list contains a sample of NEWPW operands along with their default values (if any):

MIN

This determines the minimum password length. The default is MIN=4.

MINDAYS

This determines the minimum number of days between password changes. The default is MINDAYS=1.

WARN

This determines the number of days prior to password expiration that the user is notified of the date the password expires. The default is WARN=3

NR

This indicates the number of repeating pairs allowed (for example, rabbit has one repeating pair-bb-while AABBCC has three) permited. The default is NR=0, or no repeating characters.

RS

This prevents the use of passwords from a restricted list. For information on how to design a list of restricted passwords, see your Implementation Guides.

RS is in effect by default.

ID

Specifies that the new password can not contain the ACID or parts of the name field. For example, John Smith whose ACID is USER01 cannot use “John,” “Smith” or “USER01” as part of his password. ID is in effect by default.

TS

Prevents user from specifying a password that is too similar to his previous password. For example “mean” would be too similar to “lean.” TS is in effect by default.

The remaining operands in this list are not in effect by default and must be explicitly specified if they are to be included:

NM

Specifies that only numbers can be used for passwords.

NO

Specifies that only MIN and MINDAYS will apply to new passwords.

NU

Specifies that users cannot change their passwords.

NV

Specifies that no vowels may be used in the new password.

RN

Specifies that new passwords will be randomly generated by CA Top Secret.

SW

Specifies that the new password must contain a national character (@,#,$,{,},|).

MASK

Dictates what type of character are accepted for each position of a password. The following values are used for a mask: .in +5

- c (consonant)
- v (vowel)
- n (numeric)
- x (non-vowel)
- ? (any character)

For example, MASK=cvc???n, tells CA Top Secret that all new passwords must have:

- A consonant as the first character
- A vowel as the second character
- A consonant as the third character
- Any alpha-numeric or special symbol for positions 4 to 6
- A numeric for the seventh and final character

The password “MAT2B@3” would correspond to this mask.

Note: Since password masking reduces the number of allowable passwords, it is not a recommended option.

Determining Violation Logging

The LOG option is used to designate whether or not a job/session will be logged and where the information will be sent. The LOG operands include:

SMF

Events will be written to SMF and the Audit Tracking File. This is a default for LOG. See the “Monitoring Your Security Environment” chapter for information about the Audit Tracking File (ATF).

SEC9

Violation messages will be routed to security consoles. This is also a default for LOG.

MSG

Violation messages for batch jobs, started tasks or at online user terminal. MSG is in effect by default.

INIT

Logs all job/session initiations and terminations. INIT is in effect by default.

ACTIVITY

Logs all activity for all facilities. ACTIVITY is optional.

ACCESS

Logs all resource accesses. ACCESS is optional.

NONE

Deactivates SMF and ATF logging. NONE is optional.

Note: CA Top Secret will always record audited events and violations in the Audit/Tracking File, regardless of a LOG(NONE) specification.

ALL

Selects all LOG options for all facilities. ALL is optional.

Setting the Violation Threshold

The VTHRESH options determine both the number of violations that an ACID can incur and the action that CA Top Secret will take when that violation is exceeded. The default is

```
VTHRESH(5,NOT)
```

which tells CA Top Secret, after 5 violations have occurred, to NOTify the console and the terminal through a message indicating that the threshold has been reached. Any number from 0 to 254 can be supplied in place of 5. Other options include:

CANcel

Specifies that, after the designated number of violations, the TSO or VM session, or batch job will be cancelled through the MVS recovery and termination manager (RTM) or CP FORCE command.

SUSpend

Specifies that, after the designated number of violations, the session will be cancelled and the ACID will be suspended.

RESet

Specifies that, after the designated number of violations, the SUSpend or CANcel options will be reset to NOTify

SUS/CAN,WARN

Specifies that, after the designated number of violations, CANcel and SUSpend will be enforced in WARN mode.

Preventing Password Guessing

The PTHRESH option determines the number of incorrect passwords that can be supplied before the ACID will be suspended. The default is PTHRESH(3), which means that the ACID will only be permitted 3 incorrect password guesses before he is suspended.

Chapter 7: Administering Facility Security

This section contains the following topics:

Expanding Security

Your security environment can be expanded to extend CA Top Secret security across:

- Facilities (such as TSO, CICS, VM, and others)
- Platforms (such as VSE, MVS, and VM shared environments, as well as DB2 and PC environments)
- VTAM-connected nodes (to synchronize ordinarily separate Security Files at the local and remote operating nodes) using the Command Propagation Facility (CPF)
- Other products (both CA and OEM)

All of these modifications are administered through control options.

Note: CA has made a commitment to support all IBM releases when they are in General Availability status. These products include, but are not limited to:

- CICS
- IMS
- DB2

Protecting Facilities

A *facility* in CA Top Secret is defined as an MVS subsystem that processes work on behalf of users and jobs. Two prime examples of MVS facilities are TSO and CICS, but CA Top Secret also treats VM as a facility.

In addition to predefined resource classes (kept in the Resource Descriptor Table), CA Top Secret comes equipped with a set of predefined facilities. A list of these facilities, along with their particular attributes (for example, ID, security mode) is kept in the Facilities Matrix Table (similar to how resource class information is kept in the RDT). A number of these predefined entries (the exact number depends on whether you are using MVS or VM) are called *dummy facilities* and are to be used for site-defined facilities, such as additional test regions for CICS or IMS. Each of these dummy facilities is given the USER*nn* name (*nn* is a numeric character).

The Facilities Matrix Table is administered through the FACILITY control option and suboptions. By using the TSS MODIFY command, the Security Administrator can:

- Modify existing facility entries
- Add new facilities by tailoring the USER*nn* entries

Interpreting the Facility Matrix Table

The Facility Matrix Table is stored in the Parameter File. To display the contents of the entire Facility Matrix Table or the attributes associated with a single facility within that table, use the TSS MODIFY command. For example,

```
TSS MODIFY STATUS
```

would display information for all the facilities. If you only wanted to display information for the TSO Facility, you would enter:

```
TSS MODIFY(FAC(TSO))
```

By default, the TSO Facility Matrix entry looks like the following:

```
TSO

INITPGM=IKJEFLC  ID=T  TYPE=03
ATTRIBUTES=IN-USE,ACTIVE,SHRPRF,NOASUBM,ABEND,SUAS,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,RNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,TSOC,LCFCMD
ATTRIBUTES=NOTRACE,NODORMPW,NONPWR,MSGLC
MODE=FAIL  LOGGING=INIT,MSG,SEC9
UIDACID=8  LOCKTIME=000  DEFACID=*NONE*  KEY=8
```

The first line of text in the example identifies TSO to CA Top Secret. The remaining lines specify which FACILITY operands pertain to the TSO facility within the CA Top Secret security environment; that is, how CA Top Secret will treat users signed on to the TSO facility.

FACILITY Operands

The following list of terms describes the Facility operands listed in the above TSO example

ACTIVE

Indicates that TSO is active. Opposite is INACTIVE.

SHRPRF

Allows a copy of a profile to be shared by all users in a multi-user facility (such as Advantage CA-IDMS, Advantage CA-Roscoe, IMS and CICS), which lets you conserve storage space. Opposite is NOSHRPF

NOASUBM

Indicates that authorized job submission is not being used for this particular facility. Opposite is ASUBM.

ABEND

Pertains to the region abends as the result of user violations. ABEND resets the NOABEND option. If NOABEND had been set, CA Top Secret would not cancel a user's activity in the region, even if the violations exceeded the threshold (VTHRESH). Instead, CA Top Secret would lock the user's terminal. This applies to multi-user address space facilities only.

SUAS

Single User Address Space. Requests data sets directly from MVS. In addition to TSO, other facilities fitting this description include: BATCH and STC. The functional opposite is MULTIUSER and in this case, security is generally not handled by MVS. This would include: CICS, IMS, Advantage CA-Roscoe, and Advantage CA-IDMS.

NOXDEF

Indicates that transactions and commands do not have to be authorized through LCF before they can be used. If XDEF had been specified, all commands and transactions must be defined to each user or profile through the Limited Command Facility (LCF) before the command or transaction could be used.

LUMSG

Displays the last used message. Opposite is NOLUMSG.

STMSG

Requests that the status message be displayed when a user signs on to the facility (in this case, TSO) Opposite is NOSTMSG.

SIGN(M)

Allows simultaneous logons with the same ACID for the specified facility. SIGNs disallows simultaneous logons.

Note: If there are multiple regions running under one facility (such as a CICSTESTA and CICSTESTB) an ACID can only signon once per region.

INSTDATA

Allows installation data to be stored within a TSO region. The NOINSTDATA operand can be specified to conserve storage.

RNDPW

Allows random password generation. Opposite is NORNDPW.

AUTHINIT

Resets the NOAUTHINIT operand which issues RACINIT in a problem state. See your Implementation Guide for more information.

NOPROMPT

This operand only applies to the TSO facility and pertains to password prompting during signon. NOPROMPT deactivates the PROMPT operand. If a user enters his password and user ID (ACID) at the same time (which can cause the password to be displayed), CA Top Secret will issue a warning message and temporarily lock that user's terminal before prompting for the password.

NOAUDIT

Resets the AUDIT operand. The AUDIT operand causes automatic auditing of all activity within this facility.

RES

Allows data set and volume access authorizations to be stored within the online multi-user region. NORES can be specified to conserve storage

WARNPW

Forces defined users and jobs to use their correct password during WARN mode. NOWARNPW cancels this operand.

TSOC

Indicates that a facility is TSO compatible and can handle TGET and TPUT SVCs. NOTSOC cancels this operand.

LCFCMD

Specifies that all LCF associated messages will see commands in their texts. If LCFTRANS had been specified, all LCF associated messages would then see transactions. See the “Securing Resources” chapter for an explanation of transactions and commands within LCF.

NOTRACE

Deactivates the TRACE operand. If TRACE had been specified, a diagnostic trace would have been implemented throughout the facility.

NODORMPW

Indicates that users do not need to supply a valid password in DORMANT mode. The opposite is DORMPW.

NONPWR

Does not force user to re-enter a new password for verification. This operand only applies to the TSO and CICS facilities. The NPWR operand requires password reverification.

MSGLC

Violation messages will be printed in mixed case. NOMSGLC indicates that only upper case will be used.

MODE=

Indicates what security mode will be in operation for this particular facility (not necessarily the entire installation). In this case, the TSO facility is in FAIL mode.

Note: By default, all facilities are placed in FAIL mode. You can also specify DORM, IMPL, or WARN. See the “Securing Resources” chapter for more information about security mode settings and implications.

LOGGING=

Indicates what types of security events will be recorded for this facility and where they will be recorded. The LOG control option was discussed in the “Administering Your Security Environment” chapter and applies to the entire installation. The FACILITY LOG operand applies only to a particular facility at a time. For more information about logging options, see the “Monitoring Your Security Environment” chapter.

UIDACID=

Specifies the number of characters (in this example, 8) of an online userid that will be used to derive the user's ACID.

LOCKTIME

Indicates how many minutes of inactivity can elapse for a terminal signed on to a specific facility before that terminal will be locked. Facility-specific locktimes are overridden by a User's or Profile's designated locktime. In this case, LOCKTIME is not specified.

DEFACID=

Indicates that CA Top Secret should derive the default ACID from the terminal or batch reader name if the userid entered at signon is not defined as an ACID, or if a batch ACID is not supplied. In this case, *NONE* is specified because DEFACID does not apply to the TSO facility. In the event DEFACID applies, you would have to supply a reader or terminal name.

KEY=

Indicates the TCB protect key that the facility uses for storage. In this case the default, 8, is supplied.

There are many more operands for the FACILITY control option; most notably is the case of CICS. For more information about these additional operands and suboptions, see Control Options Guide, but for now, we're going to look at how the Facility Matrix Table is maintained and how these suboption values can be changed.

Administering the Facility Matrix Table

Values contained in the Facility Matrix Table can be changed as well as displayed by using the TSS Modify command. You would use the following syntax:

```
TSS MODIFY(FAC(suboption=operand=value))
```

substituting the facility name for “suboption.”

Note: Quotes (“) are not required.

For example, to change the mode value for the BATCH facility from FAIL to WARN, you would enter:

```
TSS MODIFY(FACILITY(BATCH=MODE=WARN))
```

If you wanted to permit users to signon to the CICSTEST facility you would enter:

```
TSS MODIFY(FAC(CICSTEST=ACTIVE))
```

A list of the defaults for each facility is included in your separate *Implementation Guide*, as well as in your *Control Options Guide* under the FACILITY control option heading.

The next few sections of this chapter will introduce you to additional facility-specific security protection provided by CA Top Secret, but first, we should explain the difference between single and multiple user address space.

Single Versus Multiple User Address Space Systems

Under TSO each signed on user gets his own address space, hence, single user address space. On the other hand, under CICS, for example, each user signed on to a CICS region is given part of the region's block of space, hence, multiple user address space. Moreover, when a user requests access, standard CICS will only identify to MVS the Security Record for the region involved, not the specific user's Security Record. This procedure is typical of multiple user address space systems.

See your *Implementation: CICS* and *Implementation: TSO* guides for a more detailed description of single and multiple user address space.

MVS Online Facilities

This section details the following MVS online facilities:

- CICS Security Features
- TSO Security Features
- Advantage CA-IDMS Security Features
- Advantage CA-Roscoe Security Features
- IMS Security Features
- BATCH Security Features
- Data Lookaside Facility Security
- STC Security Features

CICS Security Features

CA Top Secret offers extensive and flexible security protection for CICS regions, transactions, and resources. No modifications to CICS code are required, therefore, CA Top Secret will remain compatible with future releases of CICS. Multi-Region Option (MRO) and Inter-System Communications (ISC) are fully supported.

In addition to the special CICS security features described here, many of the standard CA Top Secret security mechanisms can be applied to CICS. The most useful include:

- Signon Security
- Password Validation
- Terminal Security
- Authorization Restrictions
- Job Submission Validation

- Security Administration
- Security Key
- Multiple Signons

Signon Security

A user can be defined to CA Top Secret so that he must be authorized to access CICS to sign on to CICS. He can also be restricted from signing on to more than one terminal at a time. CICS signon is accomplished using the CICS CSSN transaction menu. The menu provides for entering the user's ACID name, password, and optional new password.

CA Top Secret signon processing will also operate correctly if your installation employs a non-standard signon procedure, provided the CICS signon program, DFHSNP, is ultimately invoked by that procedure.

Password Validation

All the standard password security options described in the “Identifying Users” chapter can be used with CICS.

Terminal Security

CA Top Secret can be used to restrict the use of terminals to authorized users only. Furthermore, users can be restricted to signing on only from specific terminal/readers or accessing only particular CICS regions from certain terminals.

Unattended terminals can be protected against unauthorized access through automatic terminal locking. Moreover, cumulative security violation thresholds can be established that will automatically force terminal locking if this threshold is exceeded.

Authorization Restrictions

The standard set of authorization restrictions including expiration date, time of day and day of week, CPU, and facility (in this case, CICS) can also be employed.

Job Submission Validation

Online jobs, submitted under CICS, are treated by CA Top Secret exactly as if they were submitted under TSO. See the “Identifying Users” chapter for a description of online job submission.

Security Administration

Security Administrators can use the TSS command under CICS and other online systems to perform all security administration. The syntax of the command is identical to that used under TSO or any other facility.

Security Key

A user's CICS Security Key may be specified through a TSS CREATE/ADD SCTYKEY entry. Up to 64 security keys may be specified using the SCTYKEY keyword in the CREATE or ADD command. Security keys are available in all modes.

Multiple Signons

CA Top Secret security for CICS supports multiple concurrent signons by an ACID. This is a site selectable option.

Changes to the security database made through a TSS command are immediately recognized by all facilities. Thus, a user's TSO access could be administered during a CICS terminal session.

For more information see the *Implementation: CICS Guide*.

TSO Security Features

CA Top Secret provides comprehensive security capabilities without modifications to TSO. Many of the standard CA Top Secret security mechanisms can be applied to TSO such as:

- Signon Security
- Password Validation
- Terminal Security
- Monitor Security
- Command Security
- Data Set Access Validation

Signon Security

When signing on to TSO, the user may have a TSO UADS userid defined through the TSO ACCOUNT command, however, he must also have an CA Top Secret ACID. That ACID is limited to seven characters and must match the TSO UADS userid. The TSO UADS userid is not a requirement.

Users who have been given access to TSO (through the FACILITY keyword), will be permitted to all TSO commands by default.

Password Validation

All standard password security options described in the “Identifying Users” chapter can be used with TSO.

Terminal Security

The terminal locking feature, discussed in the “Identifying Users” chapter, is also available for TSO; however, locking is enforced whenever a command (not subcommand) is about to be executed. If the amount of time since the last command execution exceeds the LTIME threshold for the user or the LOCKTIME of the facility, the terminal becomes locked.

Monitor Security

CA Top Secret provides a utility which can be executed under TSO called TSSTRACK. TSSTRACK allows administrators to monitor security-related events for one or more systems, in a realtime manner.

Command Security

TSO commands execute specific programs which need to be protected by CA Top Secret. To insure that an ACID does not gain access to a TSO command, establish ownership by adding the program to another ACID. At this point, additional users who want access to the program must be authorized to do so.

Data Set Access Validation

TSO users do not automatically have access to data sets starting with their ACID. The Security Administrator must establish ownership by adding the data set prefix to an ACID and then permitting it to the user. This can be facilitated by using masking.

For detailed information about TSO security features, see the *Implementation: TSO Guide*.

Advantage CA-IDMS Security Features

CA Top Secret provides comprehensive security capabilities in conjunction with Advantage CA-IDMS security to effectively extend resource control within the CA-IDMS environment. The key areas in which CA Top Secret provides protection for Advantage CA-IDMS are:

- Signon Security
- Transaction Security
- Password Validation
- Program and Subschema Security
- Area Security
- Terminal Security
- Job Submission Validation

Signon Security

CA Top Secret requires all users to be identified. Once the user is defined to CA Top Secret, he can use Advantage CA-IDMS.

All users must be defined to Advantage CA-IDMS in the Advantage CA-IDMS Data Dictionary, in addition to being defined to CA Top Secret. In addition, if a user is defined in the Data Dictionary as having a password, then the user will be prompted twice for a password when he attempts to sign on-once by Advantage CA-IDMS for the password in the Data Dictionary, and once by CA Top Secret for the CA Top Secret password.

Note: We strongly suggest that users be defined in the Advantage CA-IDMS Data Dictionary as not having a password, thus allowing CA Top Secret to perform all password checking.

See the *Implementation: Advantage CA-IDMS Guide* for more information on password controls for Advantage CA-IDMS signon security.

Transaction Security

CA Top Secret secures CA-IDMS transactions in two ways: protecting them at the resource level as owned resources through the OTRAN (resource class), or on a user-by-facility basis through the use of the Limited Command Facility (LCF). Both methods of security are discussed in the “Securing Resources” chapter. Considerations for administering LCF and OTRAN in an Advantage CA-IDMS environment are discussed in the *Implementation: CA-IDMS Guide*.

Password Validation

All the standard password security options described in the “Identifying Users” chapter can be used with Advantage CA-IDMS.

Program and Subschema Security

As with any resource or attribute, ownership of a program or a subschema will immediately protect the resource across all facilities and regions. Access to the program and subschema must be granted as well using the TSS PERMIT command, and limited to only specific regions through the FACILITY keyword as part of the program and subschema permission.

Area Security

Area security is provided for both logical and physical databases. Ownership of an area will protect the resource across all defined Advantage CA-IDMS regions. Access is granted using the TSS PERMIT command. Use of the area may be limited to specific regions through the FACILITY keyword, as part of the area definitions.

Terminal Security

Terminals are owned resources. Therefore, auditing and distributed administration may easily be performed on terminals. Ownership of a terminal will protect the resource across all defined facilities; however, access may be limited to only specific facilities through the FACILITY keyword, as part of the terminal definition. The name defined will depend on your environment.

Security Under Release 12.0

The CA Top Secret Advantage CA-IDMS Interface for Release 12.0 is invoked through the -CA-CIS CAISSF component. This is automatically invoked when external security is specified for the Advantage CA-IDMS Signon Resource Type Table (SRTT). CA Top Secret provides security for all Advantage CA-IDMS resources that are coded in the SRTT.

For more information on providing security for CA-IDMS Release 12.0, see your *CA Top Secret/MVS Implementation: CA-IDMS Guide* and your *Advantage CA-IDMS Security Administrator's Guide*.

Advantage CA-Roscoe Security Features

CA Top Secret provides extensive and flexible security protection for Advantage CA-Roscoe commands, monitors, and resources. Advantage CA-Roscoe versions beginning with Release 5.5 are fully supported. No modifications of Advantage CA-Roscoe code are required to implement CA Top Secret security support. Support is provided through Advantage CA-Roscoe security exits, with the code provided by Computer Associates. Among the key areas in which CA-Top Secret provides security protection for Advantage CA-Roscoe are:

- Signon Security
- Password Validation
- Terminal Security
- Data Set Access Validation
- Command and Monitor Security
- ZAP, UTILITY, IMPORT, EXPORT Security
- Job Submission Validation

Signon Security

Users must signon to Advantage CA-Roscoe using their ACID and password. To signon the user must be defined to CA Top Secret and authorized to access the Advantage CA-Roscoe facility, as well as being defined in the ROSCOE User Profiles Dataset. A user can be restricted from signing on to more than one terminal at a time.

If the Advantage CA-Roscoe key has "." imbedded in it (for example, RCA.USER5) CA Top Secret will use the characters after the period (in this case, USER5).

Password Validation

With the exception of Random Password Generation, all the standard password security options can be used with Advantage CA-Roscoe.

Terminal Security

CA Top Secret can be used to restrict the use of terminals to authorized users only.

With the installation of a command exit, automatic terminal locking becomes available. This option lets you designate a interval of inactivity, after which time unattended terminals will be automatically locked. This protects them against unauthorized access.

Command and Monitor Security

Security for both commands and monitors is provided through CA Top Secret's Limited Command Facility (LCF). Essentially, with LCF each user can have an inclusive list, which specifies a list of commands and monitors he is allowed to use, or an exclusive list which specifies a list of commands and monitors he is not allowed to use.

ZAP, UTILITY, IMPORT, EXPORT Security

Access to O/S data sets by these monitors can be closely restricted by CA Top Secret's I/O access level feature. The following access level authorizations are required to be authorized for the corresponding functions:

- ZAP (requires READ)
- ZAP with REP or SETSSI (requires UPDATE)
- IMPORT (requires READ)
- EXPORT (requires UPDATE)
- UTILITY (varies but UPDATE is common)

Job Submission Validation

Online jobs submitted under Advantage CA-Roscoe are treated by CA Top Secret exactly as if they were submitted under TSO.

Security Administration

Security Administrators can use the TSS command under Advantage CA-Roscoe to perform all security administration. The syntax of the command is identical to that used under any online facility.

IMS Security Features

CA Top Secret provides extensive and flexible security protection for IMS transactions, applications, resources, and regions in the following IMS/DC environments:

- Message Processing Regions (MPP)
- Batch Message Regions (BMP)
- Fast Path Regions (FP)

Because IMS utilizes MVS's Standard Security Interface to drive CA Top Secret, CA Top Secret is completely compatible with IMS. No modifications to IMS or IMS installation exits are required. Some parameter changes to the plain vanilla IMS security macro, however, are required to establish what is to be protected by CA Top Secret security and what is to be protected by IMS default security (SMU). For example, CA Top Secret only protects IMS commands for IMS release 4.1 and above. Therefore, for prior releases of IMS, SMU security must continue to be used for command security.

In addition to special IMS security features, many of the standard CA Top Secret security mechanisms apply. Among the most useful are:

- Signon Security
- Password Validation
- Terminal Security
- Authorization Restrictions
- Job Submission Validation
- Security Administration
- Multiple Signons

Signon Security

Most users will have to explicitly sign on to IMS, using the IMS /SIGN command. CA Top Secret checks that the user is authorized to access IMS. A user can be restricted from signing on to more than one terminal at a time.

Password Validation

All the standard password security options can be used with IMS. For sensitive transactions, password reverification can be enforced.

Terminal Security

CA Top Secret can be used to restrict the use of terminals to authorized users only. Furthermore, users can be restricted to accessing particular IMS regions from only certain terminals.

Authorization Restrictions

The standard set of authorization restrictions including expiration date, time of day and day of week, CPU, and facility (in this case, IMS region) can also be employed.

Job Submission Validation

Online jobs submitted under IMS are treated the same as for any other online facility.

Security Administration

Security administrators can use the TSS command under IMS to perform all security administration. The syntax of the command is identical to that used under TSO or any other facility.

Multiple Signons

Multiple concurrent signons by ACIDs is a site selectable option.

Changes to the security database made through a TSS command are immediately recognized by all facilities. Thus, Advantage CA-Roscoe can be administered through an IMS session.

BATCH Security Features

CA Top Secret also provides security protection for the BATCH facility. The key areas in which CA Top Secret protects the BATCH facility are:

- Signon Security
- Password Validation
- Card and Remote Reader Security
- Job Submission Validation
- Job Scheduling Security

Signon Security

CA Top Secret also views BATCH as a facility that must be protected and authorized for use. To provide this protection, each batch job must be associated with an ACID and password. CA Top Secret treats a batch job exactly like an ACID, such that it has an associated user record with a set of specific access authorization.

A description of the methods used to identify the ACID associated with a batch job can be found in the *Implementation: BATCH and STC Guide*.

Password Validation

In IMPL and FAIL mode, a user must supply a valid password. In WARN and DORM mode, you can use the FACILITY control option to force the user to supply a valid password. The requirements for password validation for different modes are discussed in the *Implementation: BATCH and STC Guide*.

Card and Remote Reader Security

For jobs being submitted from a physical reader, RJE, or NJE terminals, the submitter's password must be manually coded in the PASSWORD= parameter on the JOB statement, unless the associated ACID does not require a password.

Job Submission Validation

By default, CA Top Secret allows a defined user to submit batch jobs for which the ACID is specifically authorized. Explicit authority is required to permit a user to submit jobs identified by other ACIDs. The required authority needed to submit these jobs is granted through the ACID keyword of the TSS PERMIT command.

Job Scheduling Security

Production Job Scheduling Systems (such as Unicenter® CA-Scheduler® Job Management) may by given full authority to submit any job as required. This authority can be:

- Limited to a specified list of jobs
- Restricted by facility
- Restricted by privileged program

Data Lookaside Facility Security

Under release 3.1.3 and above of MVS/ESA, an additional feature called the Data Lookaside Facility (DLF) was added to control the loading of selected data sets into ESA hiperspaces by selected jobs. With the proper authority and keywords, CA Top Secret can identify and control those data sets and jobs, which are eligible for DLF.

For a description of the DLF Record (which maintains a list of data sets and jobnames eligible for DLF processing), the associated keywords, and the authority needed to maintain this record, see the *Command Functions Guide*.

See the *Implementation: BATCH and STC Guide* for an in-depth discussion of all BATCH security features.

STC Security Features

CA Top Secret offers security protection for all required STC definitions and STCs which reference sensitive data or affect system integrity. In addition, you have the option to protect whatever is appropriate at your particular installation. CA Top Secret provides the flexibility to adhere to the scope or degree of STC protection by providing four levels of security:

- Operator Accountability
- ACID Association
- Password Security
- Bypass Security Checking

Operator Accountability

Data processing personnel continuously have access to the O/S consoles and started tasks which mean that many STCs may be executed without any record to indicate who actually entered the started task.

With the proper authority, CA Top Secret allows a Security Administrator to force the operator executing the STC to provide identification. By using the ADDTO(STC), the administrator can attach an STCACT attribute to the STC definition, forcing the operator to enter his user ACID and password. If the ACID or password entered is invalid, the STC does not execute.

ACID Association

An STC is associated with a specific ACID by issuing the TSS CREATE and TSS ADDTO commands. Specifying a password is required only for critical STCs.

Password Security

You can define an STC that prompts for an ACID with an assigned password. If the ACID assigned to the STC is defined with a password, then a second prompt is issued for the password. This option provides an additional form of protection by forcing the operator to supply the correct ACID information for the STC before it is allowed to execute.

Bypass Security Checking

To define a specific STC that bypasses CA Top Secret security checking, use the TSS ADDTO command and specify the BYPASS keyword in the acidname field. This allows the started task to execute without any security checking.

For a detailed description on Started Task security for defined and undefined STCs, as well as Security options for undefined STCs, see the *Implementation: BATCH and STC Guide*.

VM as a Facility

As supplied, virtual machines running CA Top Secret are doing so under the VM facility. CA Top Secret controls access to the VM facility by requiring that the user be authorized to use the virtual machine. By default, only the MSCA is authorized to use VM when CA Top Secret is first installed. Everyone else must be explicitly authorized to use the VM facility through a TSS CREATE or ADDTO function. If you want to segregate your VM CPUs into different facilities use the FACILITY control option to rename one of the dummy USER*nn* entries in the Facility Matrix Table. For example:

```
FACILITY(USER1=NAME=VMTEST)
```

designates the USER1 entry as VMTEST, and

```
FACILITY(VMTEST=MODE=WARN)
```

sets the mode to WARN.

Next, use the VMFAC control option to associate your CA Top Secret facility to the HCPSYSID of the CPU by entering:

```
VMFAC(SYSTEMC=VMTEST)
```

In this example, SYSTEMC identifies the SYSID for HCPSYSID.

As you can see, CA Top Secret is flexible enough to be worked into a production VM environment. Just as you can set up a test CPU, you can also set up other test and production CPUs.

VM Facility Deactivation

Security Administrators (who have the correct authority) can deactivate the VM facility by using the TSS MODIFY command to specify the INACT (inactive) suboption of the FACILITY control option. The following example prevents users from logging on to VMTEST.

```
TSS MODIFY FAC(VMTEST=INACT)
```

The facility may be reactivated in the same manner using the ACTIVE suboption of FACILITY: The following example allows the user to sign on to the VMTEST facility.

```
TSS MODIFY FAC(VMTEST=ACTIVE)
```

Sharing Security Files between VSE, MVS, and VM

CA Top Secret provides the capability to share Security Files between its VSE, MVS, and VM products. This feature is optional and can be implemented at any time.

The advantages for sharing your Security File include:

- More efficient control of your security environment
- The flexibility to administer security from the MVS or VM system
- The ability to maintain one Security Record for each ACID, making it easier to gather user information
- The ability to gather information about resources that are accessible from VSE, MVS, and VM

The following requirements must be met to ensure that this feature functions successfully:

- Releases for VSE, MVS, and VM must be compatible (for example, CA Top Secret MVS 5.0 and CA Top Secret OS/390 5.1 can share Security Files with CA Top Secret VM 1.4, but cannot with the CA Top Secret VM 1.1 and earlier).
- Encryption keys provided at installation must be identical. See the *Getting Started* for CA Top Secret VSE, MVS, and VM for information on the encryption key.
- The SECFILE control option for CA Top Secret VM must have a data set name that matches the data set specified on the SECFILE DD statement for the CA Top Secret MVS Started Task.
- The SHRFILE control option for CA Top Secret VSE, MVS, and VM must be set to YES or SECURITY, so that the physical lock record (located in the Security and/or Audit files) is accessible to all systems sharing the Security File.

  Note: VSE, MVS, and VM share one Security File but each has a separate Parameter File which stores these control options.
- In the VM directory for the CA Top Secret Server Machine you must establish a multi-write link to the VSE or MVS volume which holds the Security File.

Chapter 8: Implementing Ownable Resource Security

This section contains the following topics:

Resource Types

With CA Top Secret, you can protect logons, as well as ownable CA Top Secret resources such as the CP DIAL command, minidisks, VM readers, DCSSs, RSCS nodes, CP commands, diagnose codes, IUCV, VMCF, OS/DOS data sets, DASD volumes, and CPUs. This chapter also discusses how installation-defined resources can be either ownable or unownable.

Implementing LOGON Protection

Without CA Top Secret, any user who logs on to VM must be listed in the VM directory. However, by putting CA Top Secret security in place, you can establish additional points of control for the user to log on.

First, define the user to CA Top Secret. In doing so, you have many options. The following is an example of one such definition:

```
TSS ADDTO(USERA) SOURCE(terminal) PAS(password) FAC(VM)
TSS PERMIT(USERA) CPU(cpu) TIME(08,16) DAYS(M,W,F)
```

Setting up USERA as in the example above means that the user can only log onto VM from a certain terminal on a certain CPU between 0800 and 1600 hours on Monday, Wednesday, and Friday. Furthermore, the user must supply the correct password to get access to any of these resources.

When CA Top Secret security is operating, all users are asked to supply logon information through the following prompt (even when the CA Top Secret password is not enforced):

```
"TSS0100A Enter password, LOGOFF, or HELP (it will not appear when typed):"
```

When CA Top Secret security is NOT operating, an IBM prompt appears. Instead of the CA Top Secret password, all users must supply the password from the CP directory.

Note: At logon, the minidisks that are automatically linked must be verified by CA Top Secret. To be verified, they must appear in that user's Security Record, as they appear in the CP directory. For instance, if the CP directory has USERA set up for automatic access to MAINT.0190, MAINT.019D, and MAINT.019E, the Security Administrator must authorize USERA for access to those minidisks:

```
TSS PER(USERA) VMMDISK(MAINT.0190,MAINT.019D,MAINT.019E) ACC(READ)
```

If the Security Administrator fails to authorize the user for access to the correct minidisks at LOGON and the user is in non-DORMANT mode, the user receives a CA Top Secret violation message. Therefore, be sure that the Security Administrator matches the CP directory entry exactly. That includes duplicating the access level(s) given in the CP directory.

Implementing VMDIAL Protection

VMDIAL provides security for DIAL-able logical terminals. For instance, suppose USERJOE wants to issue a command, in order to access TSO (which is running under MVS on a virtual machine):

```
CP DIAL MVS 82
```

The security administrator must be sure that only qualified users dial the MVS machine. Thus, the administrator first adds this access authorization to a department:

```
TSS ADDTO(DEPT001) VMDIAL(MVS)
```

Note: In the example, the ownership of the MVS virtual machine's DIAL-able GRAF devices are added to a department. You can do the same thing for a division or zone.

Then, because the user is within the scope of that department (or attached to a profile), the user can be authorized to issue DIAL commands:

```
TSS PERMIT(USERJOE) VMDIAL(MVS)
```

Now, when USERJOE issues a DIAL MVS 82, the user is asked for the ACID and password. By supplying the correct ACID and password, the user is allowed to DIAL directly into the 0082 terminal port.

Note: By using the method just described, the user can implement DIAL protection even for users who are not identified in the CP directory. Because the user accesses MVS directly without first logging onto a virtual machine, the user does not need to be known to VM. However, the user needs FAC(VM) or the facility defined for the terminal from which the user is issuing the DIAL.

Implementing Minidisk Protection

Minidisks can be accessed through MDISK (minidisk) or LINK statements in the VM directory, or with the CP LINK command.

No authorization is required for directory MDISK statements. As you may know, VM recognizes a user's right to access the minidisks because they are defined in MDISK statements in the CP directory. Therefore, when a user links to the minidisk, VM processes the request. The same thing holds true for CA Top Secret security. However, when one user tries to LINK to another user, CA Top Secret intervenes.

Users gain access to minidisks at logon. Therefore, as mentioned above, administrators must be careful to match CA Top Secret authorizations with VM directory or CP LINK statements. To do so, a Security Administrator can use a TSS PERMIT command:

```
TSS PERMIT(DCADEPT1) VMMDISK(USER01.,USER02.,USER03.) ACCESS(READ)
```

That PERMIT allows the DCA to link to and access (READ) USER01's, USER02's, and USER03's minidisks.

Minidisk LINK access, under CA Top Secret control, includes the following access levels, which are defined in the RDT Record:

- READ SMULTI
- WRITE EREAD
- MULTI EWRITE
- MWRITE UPDATE

- MREAD ALL
- SREAD NONE
- SWRITE

Since minidisks are protected by default when CA Top Secret is in FAIL mode, there is no need to set up default protection for them. However, the Security Administrator may, at some point, need to change some other aspect--such as the allowable access levels (ACLST) or the default access level (DEFACC) of minidisks.

To modify the access levels for minidisks, the administrator must have the proper MISC1 authority. Then, the administrator can use this TSS command syntax:

```
TSS REP(RDT) RESOURCE(VMMDISK) ACLST(new access levels)
   DEFACC(new default access level)
```

VM Reader Protection

The VMRDR keyword lets administrators control who can put files into whose virtual reader. For instance, the CMSBATCH virtual machine's reader might otherwise be unprotected from a security standpoint.

For example, the following command prevents a user from punching a file to CMSBATCH:

```
TSS ADD(PROFRND) VMRDR(CMSBATCH)
```

From a security standpoint, the key to implementing VMRDR security is to remember that VMRDR lets the ACID holder place spool files in the specified userid's virtual reader.

DCSS/NSS Protection

Most often, discontiguous saved segments (DCSSs) or named saved segments (NSSs) are used to share read-only storage between two or more virtual machines. An example is a DCSS for CMS. The CMS DCSS consumes more than 300K bytes of memory. By sharing a CMS DCSS, all CMS machines can point to a single common 300K+ byte section of memory, instead of having one copy per machine.

Both shared and unshared DCSSs are protected by CA Top Secret. With CA Top Secret's DCSS keyword, you can secure standard program products such as PROFS (online message centers), SAS, and VSAM that run in a DCSS. To limit who is allowed to use these products, do not PERMIT users to use the DCSS. For example, the following lets USER82A use the SAS program, which resides in a DCSS.

```
TSS PERMIT(USER82A) DCSS(SAS)
```

Of course, the SAS segment must be defined by the system programmer in DMKSNT or via the DEFSEG command just as CMS must be defined.

Note: TSS PER(USERA) DCSS(CMS) does not protect CMS from IPLs. Instead, to protect CMS (or any other IPL-able DCSS), you must use the CPCMD keyword.

For example:

```
TSS PERMIT(USERA) CPC(IPL.CMS)
```

Using some variation of the last TSS command, you can protect IPL-able DCSSs.

RSCS Node Protection

When dealing with networks of virtual machines, the typical site uses RSCS to transfer files between virtual machines on separated real machines. The target machine is identified with the CP command TAG. In a typical TAG command, the node specifies the destination (what machine) of the job and the user of the virtual machine at the node.

Any defined RSCS nodes can be protected by CA Top Secret. For example:

```
TSS ADD(TECHDEPT) VMNODE(DALLAS)
TSS PER(USER01) VMNODE(DALLAS)
```

Allows USER01 to transfer data to the RSCS node in Dallas.

Terminal Protection

The CA Top Secret resource class keyword TERMINAL is used to refer to both local and VTAM terminals. Terminals can be secured in two ways:

- Users can be restricted from using a protected terminal.
- Users can be forced to log on only from an authorized source terminal. This type of terminal protection can be used to channel sensitive activities through authorized pathways.

Using the TERMINAL Keyword

The TERMINAL keyword is used to define terminals to the system and to establish access authorizations. If a generic prefix is used, its length must be between one to eight characters. For example, to assign ownership of VTAM terminal L1032209 to the accounting department, enter this TSS command:

```
TSS ADD(ACTDEPT) TERM(L103)
```

For examples of how to define the various types of terminals refer to *Command Functions Guide*.

Once defined to CA Top Secret, access to terminals can be specifically restricted. For example, the following authorizes USER01 to use L103 terminals in the morning.

```
TSS PER(USER01) TERM(L103) TIMES(07,12)
```

To give everyone this kind of authority enter:

```
TSS PER(ALL) TERM(L103) TIMES(07,12)
```

To allow a user to access all protected terminals enter:

```
TSS PER(acid) TERM(*ALL*)
```

Remember that TERM(*ALL*) must have previously been assigned to the MSCA:

```
TSS ADD(MSCA) TERM(*ALL*)
```

Using Source-of-Origin Security

Selected users (or profiles) can be forced to log on from a designated terminal. For example, to force USER01 to access the system only from VTAM terminal L1036119, enter a command similar to this one:

```
TSS ADDTO(USER01) SOURCE(L1036119)
```

Note that terminal L1036119 does not even have to be defined to CA Top Secret. However, if the specified terminal is defined, then be sure the user is authorized to access it:

```
TSS PERMIT(USER01) TERM(L1036119)
```

A source-of-origin restriction overrides TERMINAL authorization. That is, if the previous SOURCE function were in place but this TSS PERMIT had been issued:

```
TSS PERMIT(USER01) TERM(L1036230)
```

USER01's attempted logon from L1036230 results in a security violation because USER01 can only log on from L1036119.

Control Program (CP) Commands Protection

Specific CP commands can be protected with CA Top Secret.

For example, the following gives USER01 the ability to use the CP QUERY command.

```
TSS PERMIT(USER01) CPCMD(QUERY)
```

In CA Top Secret, CP commands are grouped into three categories:

- General user commands
- Privileged commands
- Mixed commands

General User Commands

These commands mean only one thing to all users. A "general user" command is also one that only requires a class G user status in a standard VM environment.

Examples: SPOOL, LINK, STORE, TAG, SEND

By authorizing users to these general commands, the Security Administrator gives access rights similar to rights accorded class G VM users.

Privileged commands

In VM, these commands mean only one thing to everyone. However, only a few select users are given the class that allows them to use these privileged commands.

Examples: STCP or STORE.H, SHUTDOWN, NETWORK

In a typical VM environment, these commands require a specific class, other than class G. CA Top Secret considers these commands to have implied VMPRIVilege access rights. That is, if USERA were PERMITted to the CP command SHUTDOWN, CA Top Secret makes USERA a privileged VM user--whether VM allows it or not--within the context of that command.

Mixed commands

These classes can have different meanings depending on the user's class.

Examples: SET, QUERY

Some CP commands have different meanings for different privilege classes. For instance, the class B user who types in "QUERY DASD" receives information about the real DASD associated with the real CPU. On the other hand, the class G users who type in "QUERY DASD" get information about the virtual DASD associated with their virtual machines. As you recall, CA Top Secret (in FAIL mode, and when permitted in WARN mode) supersedes VM's privilege classes. What happens when a user needs to query the DASD?

Suppose that the DCA for the Financial Department called FINDEPT gives USER01 the authority to issue the CP QUERY command:

```
TSS PERMIT(USER01) CPCMND(QUERY.DASD)
```

With this permit, USER01 may now issue a

```
QUERY DASD
```

For mixed CP commands, such as QUERY, that are either unprotected or executed in DORMANT mode, or protected but not permitted in WARN mode, the syntactical variations accepted by CP and the format or degree of their responses are determined by the privilege classes defined for the user in the CP directory.

In FAIL and IMPLEMENT modes, when permitted in WARN mode, or with ACTION(FAIL) in DORMANT mode, CA Top Secret determines the user's privilege in executing a command like QUERY.

In order to grant users privileged access to protected CP commands, you must use ACTION(VMPRIV). VMPRIV means VMPRIVilege in a PERMIT for the basic CP command or for a specific variant.

You can use a TSS command like this one:

```
TSS PERMIT(USER01) CPCMD(QUERY.DASD) ACTION(VMPRIV)
```

where:

- TSS PERMIT=command function
- USER01=ACID
- CPCMD=keyword
- QUERY.DASD=operand of CPCMD keyword

- ACTION=keyword
- VMPRIV=operand of ACTION keyword

With this last TSS command function, USER01 is permitted to use QUERY DASD as a privileged user. In turn, USER01's "QUERY DASD" returns a list of the real DASD, instead listing USER01's virtual DASD. Therefore, ACTION(VMPRIV) is CA Top Secret's way of controlling how ambiguous resource commands are used. The commands are ambiguous because they have one meaning for class G users and another meaning for privileged users. To reiterate, certain CP commands, including QUERY, can be ambiguous.

CA Top Secret also gives you the ability to limit the use of CP commands through the VMUSER keyword. The VMUSER keyword denotes who may be the target of the CP command. For example, consider the CP command FORCE.

```
TSS PER(USER01) CPCMD(FORCE)
```

Gives USER01 the ability to FORCE any virtual machine off the system.

```
TSS PER(USER01) CPCMD(FORCE) VMUSER(USER02,USER03)
```

Gives USER01 the ability to FORCE the USER02 and USER03 machines off the system, but prevents USER01 from forcing any other machine off the system.

```
TSS PER(USER01) CPCMD(FORCE) VMUSER(USER(G))
```

Gives USER01 the ability to FORCE only those machines whose name begins with USER.

See the "CP Command Attributes Table" appendix for a list of CP commands which support the VMUSER keyword.

Diagnose Code Protection

Resource DIAGNOSE protects VM diagnose codes with the following TSS command function:

```
TSS PERMIT(USERA) DIAG(84)
```

In this example, USERA has access to the "84" diagnose code. When you list the information for that ACID, you see the code in its full four-character format: 0084.

Like some CP commands, certain diagnose codes have privileged and non-privileged classes. For example:

```
TSS PER(USER01) DIAGNOSE(A0)
```

Allows USER01 access to the class 'G' form of the diagnose.

```
TSS PER(USER02) DIAGNOSE(A0) ACTION(VMPRIV)
```

Allows USER02 access to the privileged class of the diagnose.

There are several diagnose codes that are used extensively by CMS. Access to these diagnose codes is assumed unless specifically restricted. One example is diagnose code 8 (execute CP command). If that diagnose code is owned, anyone without a permit would still be allowed to execute that diagnose unless his or her ACID has been denied:

```
TSS PERMIT(useracid) DIAGNOSE(8) ACTION(DENY)
```

Below is the list of diagnose codes that are processed in the same way:

```
00, 08, 0C, 10, 18, 20, 24, 40, 54, 58, 5C, 60,
68, 70, 8C, A4, A8, B0, C8, 210, 218, 250, 260, 270
```

OS/DOS Data Set Protection

CA Top Secret does not automatically protect OS/DOS data sets except in FAIL mode. In order to give default protection to OS/DOS data sets in any other mode, you must attach the DEFPROT attribute using the TSS REPLACE(RDT) command function.

CA Top Secret protects all defined (meaning “owned”) OS/DOS data sets through the FACILITY control option SIOCHK: FAC(VM=SIOCHK)

Note: Caution should be used when initially implementing this feature. In DORMANT mode with the SIOCHK option specified, CP level OS/DOS data set protection may deny access to a resource (or fail an I/O request) if it is unable to determine the resource being accessed.

Steps to Implementing CP-level Protection

To implement CP-level protection:

- Begin with NOSIOCHK during production hours
- Identify potential complex I/O users and test
- Define ACIDs and add NODSNCHK,NOVOLCHK for: Guest operating systems, Complex I/O users, Backup systems (DDR, etc), and Data base servers
- Enable SIOCHK during production hours

DASD Volume Protection

CA Top Secret protects DASD volume access to OS- and DOS-formatted volumes. DASD volumes can be specified either by their full names or by generic prefix. If necessary, you can also bypass volume level security entirely.

Specific volumes can be protected by identifying their volume serial numbers. An example of the required syntax follows:

```
TSS ADDTO(DEPT01) VOLUME(24T921)
```

Or groups of volumes can be protected by specifying a generic prefix:

```
TSS ADDTO(DEPT01) VOLUME(24T)
```

It protects all volumes whose VOLSER begins with “24T.”

Users or profiles can be allowed access to all volumes through a simple two-step process. First define the VOLUME(*ALL*) to the Master SCA:

```
TSS ADDTO(mscaacid) VOLUME(*ALL*)
```

Next, make a CA Top Secret entry that gives the designated user, profile, or everyone access to all volumes. The level of access allowed may be restricted. For example, the following authorizes any user for READ access to any volume.

```
TSS PERMIT(ALL) VOLUME(*ALL*) ACCESS(READ)
```

Once a volume has been defined, an access request lacking sufficient OS/DOS data set authorization, must be specifically permitted.

Volume level access allows a user to access, up to the authorized level, any OS/DOS data set on the volume. For example, an access level of READ does permit a backup procedure, but not a restore procedure. Using the *ALL* reserved word, any group of users can be authorized for access to any volume--current and future.

For example, the following authorizes this profile group to access any volume at any level.

```
TSS PERMIT(OPPROF) VOL(*ALL*) ACCESS(ALL)
```

The following authorizes any user for READ access to any volume.

```
TSS PERMIT(ALL) VOLUME(*ALL*) ACCESS(READ)
```

CPU Protection

Security Administrators can use CA Top Secret to protect a specific CPU (identified by HCPSYSID or system_identifier statement in the system configuration file) once it is owned. Use the CPU keyword, along with TSS CREATE or ADDTO, to establish resource ownership.

Once owned, a CPU cannot be accessed unless explicit authorization is granted. The CPU keyword is the vehicle for specifying CPU ownership and authorizations.

For example, the following assigns ownership of this CPU to the department associated with ACID DEPT01.

```
TSS ADDTO(DEPT01) CPU(VMSYSA)
```

Remember that CPU is a “resource class” type of keyword--like CPCMD, VOLUME, or VMMDISK.

To grant USER01 access to VMSYSA, a Security Administrator with the required administrative authority enters the following:

```
TSS PERMIT(USER01) CPU(VMSYSA)
```

By qualifying a TSS PERMIT with other controls, CPU protection can be used to limit access to a CPU on a time-of-day basis. For example, to provide for a virtual machine that can be accessed by the day shift, a CA Top Secret administrator enters a TSS command similar to the following example:

```
TSS PER(DAYPROP) CPU(VMSYSA) FACILITY(VM) TIMES(09,17)
```

SFS Command and Directory Protection

CA Top Secret security for the IBM Shared File System (SFS) environment utilizes the CA-CIS CA-ESM component. CA-ESM “tells” the SFS server machine that external security is in effect. All user requests to access SFS directories and files, as well as some SFS command requests will then be redirected to the CA Top Secret server machine. The DIRECTRY and SFSCMD resource class keywords can then be used to customize ACID access to these resources.

For further information on the CA-ESM component refer to your CA-CIS documentation.

Securing SFS Directories

The DIRECTRY resource class is used to restrict user access to particular SFS directories. The syntax is as follow:

```
TSS PER(acid) DIRECTRY(directory[:subdirectory,],...)
   [FILE(filename)] [POOL(filepool)] [ACCESS(level    )]
```

directory

specifies the SFS directory name. You can optionally specify one or more subdirectories within that particular directory. Extended masking is supported using the %, *,and + characters. If you intend to use the "+" as a masking character, you should note that it is also considered a valid character for CMS directory and file names.

Up to five (5) directories can be specified per PERMIT.

FILE

specifies a particular file or file prefix within that directory. The FILE parameter is optional.

POOL

Specifies a particular file pool (SFS server machine) in which the directory is located. This parameter is also optional and can be used when there are two like named directories existing on different file pools.

ACCESS

specifies the type of access the ACID will have to that directory. DIRECTRY supports the following access level:

- CREATE
- READ
- WRITE
- UPDATE
- ALL
- NONE

If no access level is specified, READ will be assumed. UPDATE access is only supported for files that currently exist in the directory.

Note: If you have stored commonly used or accessed EXECs or files on an SFS directory, you should PERMIT that directory to the ALL Record with an access level of READ.

To better understand the use of the DIRECTRY resource class, consider the examples in the following section.

Examples - DIRECTRY

All public EXECs for the TOPS company reside in the 'POOL1:SYSTEM. EXECS.PUBLIC' filepool directory. Also within this directory are a number of EXECs used by the Production Control Staff for job scheduling, by the Applications Programmers for compiling and by the Administrative Assistants for various other job functions.

If each of these EXECs is grouped by a standardized prefix (i.e., PC for Production Control, AP for Applications Programming, AA for Administrative Assistance), the following commands would be issued to prevent these groups from accessing each other's EXECs:

For Production Control:

```
TSS PER(PCNTLP01) DIRECTRY('SYSTEM.EXECS.PUBLIC.') ACC(READ) POOL(POOL1)
TSS PER(PCNTLP01) DIRECTRY('SYSTEM.EXECS.PUBLIC.') ACC(READ) FILE(PC*.*)
   POOL(POOL1)
```

For Application Programming:

```
TSS PER(APPLPP01) DIRECTRY('SYSTEM.EXECS.PUBLIC.') ACC(READ) POOL(POOL1)
TSS PER(APPLPP01) DIRECTRY('SYSTEM.EXECS.PUBLIC.') ACC(READ) FILE(AP*.*)
   POOL(POOL1)
```

For Administrative assistants:

```
TSS PER(ADMINP01) DIRECTRY('SYSTEM.EXECS.PUBLIC.') ACC(READ) POOL(POOL1)
TSS PER(ADMINP01) DIRECTRY('SYSTEM.EXECS.PUBLIC.') ACC(READ) FILE(AA*.*)
   POOL(POOL1)
```

Securing SFS Commands

The SFSCMD resource class is used to restrict ACID access to certain SFS commands. The syntax is as follows:

```
TSS PER(acid) SFSCMD(cmd [.filepool],...)
```

cmd

Specifies the SFS command. You can specify any of the following commands:

- CONNECT
- DATASPACE
- DELETE
- DIRATTR
- DMSDISFS
- DMSDISSG

- DMSENAFS
- DMSENASG
- DMSOPCAT
- DMSQUSG
- DMSRELBK
- DMSWRACC
- ENROLL
- FILEPOOL
- MODIFY
- QUERY
- RELOCATE
- RENAME
- SMSCDRA
- SMSERASE
- SMSOPEN
- SMSOPENX

filepool

Specifies in which SFS file pool that access can occur. This parameter is optional.

Examples - SFSCMD

There are two filepools at the TOPS company POOL1 and POOL2. USER01 needs the authority the submit the ENROLL command in both file pools. USER02, however, has limited authority in POOL2 and should, therefore, only be permitted to execute the ENROLL command in POOL1. The administrator enters the following commands:

```
TSS PER(USER01) SFSCMD(ENROLL)
TSS PER(USER02) SFSCMD(ENROLL.POOL1)
```

Communication Between CA Top Secret and SFS

There are two important points you need to keep in mind regarding the exchange of security information between CA Top Secret and the SFS server machine.

- The first time a user attempts to access an SFS resource, the SFS Server machine invokes CA Top Secret and builds an internal table to maintain the security information it receives. For any subsequent access attempts, the SFS server will consult the internal table rather than CA Top Secret unless CMS is (re)IPLed.
- When the SFS server is brought up, CA Top Secret will tell the server what DOWN options are in effect. If those DOWN options subsequently change, CA Top Secret will NOT inform the server unless the SFS machine is brought down and then restarted.
- All filepool users and administrators will need the following CA Top Secret authorization:

  ```
  TSS PERMIT(ACID) VMMACH(FILEPOOL) ACCESS(APPC)
  ```

Securing Data Spaces

The DSPACE resource class is used to prevent a service machine (or any other machine) from giving access to its data spaces to an unauthorized machine. Consider the following command:

```
TSS PERMIT(USER01) DSPACE(SFSSERVR) ACCESS(READ)
```

In this example, any virtual machine running with USER01 ACID can be given READ ONLY access to a data space created by virtual machine SFSSERVR. It should be noted that, in this example, SFSSERVR is a virtual machine, not an ACID.

The actual DSPACE resource check takes place when SFSSERVR issues the ADRSPACE PERMIT macro internally to give access to a data space created by SFSSERVR. Since this action is driven by a request to SFSSERVR by USER01, any violation messages are issued to the machine requesting the action (in this case, USER01) and not to the machine actually driving the security check (SFSSERVR).

For further details on using the DSPACE resource class, including information on access levels, refer to the *Command Functions Guide*.

Installation-Defined Resources

CA Top Secret can protect installation-defined resources. Installation-defined resources are those resource classes for which VM's access control interface does not normally issue a security call, but which nevertheless, must be protected. Implementing security for such resources involves customizing CA Top Secret exits or interface calls in VM products or applications programs.

CA Top Secret provides a group of reserved keywords that reference these installation-defined resource classes:

- ABSTRACT
- UR1, UR2
- USER*xx*

The first three keywords are used for owned resource types; the last, USERxx, for unowned resource types. Ownable resources may only be restricted by expiration date, access level, time and day, privileged program, and facility.

ABSTRACT

Use ABSTRACT, along with additional site-written code, to extend security to other "system" type resources. ABSTRACT is reserved for use by CA Top Secret.

UR1 and UR2

UR1 and UR2 are resource class keywords reserved by CA Top Secret to designate owned, non-system type resources that are site-defined, such as particular account codes. They are implemented and function in the same manner as ABSTRACT.

For example:

To assign ownership of an account number, the administrator enters:

```
TSS ADDTO(CORPORAT) UR1(ACCT 987I)
```

USERxx

In addition to the pre-defined facility entries, 78 dummy facility entries with names of USER0 through USER77 are available for site customization.

For example:

To allow a user to access several unowned user resources, the administrator enters:

```
TSS ADDTO(USER05) USER9(WELL,OIL,GAS)
```

Note: All of the above-mentioned installation-defined resources are pre-defined in the RDT Record.

Any installation-defined resource class can be implemented as either ownable or unownable. The decision depends upon the particular installation's situation and requirements with respect to that resource type. If factors such as administrative scope and hierarchy are pertinent to who are allowed to grant authorization to this resource type, then it must be made ownable. Or, if it is advantageous to restrict the use of this resource type--by time of day or access levels--again, make the resource ownable. If neither of these considerations apply, make the resource unownable, since it is easier to administer.

IUCV and VMCF Security

Control over Inter User Communication Vehicle (IUCV) and Virtual Machine Communications Facility (VMCF) functions is available with CA Top Secret.

This feature allows you to control a virtual machine's ability to CONNECT with any other users and CP system services and to mediate VMCF communications between virtual machines.

```
TSS PERMIT(USERA) IUCV(USERB)
```

In this example, USERA is authorized to perform an IUCV CONNECT function with USERB as the target of the connection.

If you wish to enable IUCV CONNECT security, you should add OPTIONS(4) to the server startup Parameter File. You should also ensure that their secondary user is not connected to *MSG, or connected to any user connected to *MSG.

Making IUCV CONNECT optional was done to avoid possible system hangs if the server is processing any IUCV CONNECT requests for its secondary user, or for any ACID connected to the secondary user via *MSG.

Providing a Security Environment for VM Batch Subsystems

Assume, for example, that you wish to correct a security exposure between a batch controller machine named “BATCHC” and three batch worker machines “BATCH01,” “BATCH02,” and “BATCH03.”

To permit the controller use of the diagnose required to invoke the surrogate function, enter:

```
TSS ADDTO(dept) DIAG(D4)
TSS PER(BATCHC) DIAG(D4)
```

To permit the controller to surrogate the batch worker machines, enter one of the following:

```
TSS ADDTO(dept) VMMACH(BATCH0)
TSS PER(BATCHC) VMMACH(BATCH0) ACC(SUROGATE)
```

To permit the controller access to the job submitters’ ACIDs so that they can be used in surrogacy either by explicitly permitting the controller to the ACID of each potential submitter or through the NOSUBCHK attribute, enter one of the following:

```
TSS PER(BATCHC) ACID(submitter, submitter, etc.)
TSS ADDTO(BATCHC) NOSUBCHK
```

If your batch jobs typically link to the submitter’s minidisks, a security check will be performed when the batch worker attempts the link. The following commands will provide this permission:

```
TSS ADD(msca acid) VMMDISK(%.)
TSS PER(ALL) VMMDISK(%.) ACC(READ)
```

This will allow any ACID READ access to its own virtual machine’s minidisks.

Note: The CMS batch facility included in the VM base product does not support external security. It is possible, however, for an installation to implement surrogacy for CMS batch by defining a trusted “front-end” virtual machine. This machine would receive a submitted job, validate the submit authority using the CA Top Secret Application Interface, surrogate the CMS batch machine, and then transfer the job to it. Care must be taken in setting up this type of application to ensure proper serialization of work.

Controlling Job Submission to MVS Systems

CA Top Secret can help you to restrict user job submission to MVS guest and to JES systems on NJE networks. Specify the VMRDR resource to protect job submission to an MVS guest through virtual readers generated as “real” JES readers.

VMNODE protection can be used for networked JES systems through RSCS. VMNODE controls user job submission. You can use the VMJESLNK control option on the MVS side to provide implicit authorization for jobs submitted through this path.

Chapter 9: Batch Operations

This section contains the following topics:

Batch Jobs

CA Top Secret Batch operates within the CA Top Secret server machine. The batch task is started automatically during server initialization and continuously monitors the server's card reader for incoming jobs. Whenever a card reader interrupt occurs, the batch task searches all spool files in the server's reader queue and attempts to process all those which possess the required characteristics of a CA Top Secret batch job.

In order to be considered for batch execution, a spool file must be type "PUN" or type "RDR." Its spool file class must be "B," and it must not be held (neither SYSHOLD nor USERHOLD). CA Top Secret does not inspect the contents of a spool file until after it has been selected for potential execution. As long as the file conforms to the restrictions outlined above, it causes a batch job to be started and logged. If the file does not contain valid Job Control statements, this is detected by the Job Control Language processor during Job initiation. See the heading entitled: "Job Control Statements and Job File Format" following in this section for information on the contents of a CA Top Secret batch job file, and how to submit batch jobs to the server.

CA Top Secret Batch ignores any files in the server's reader which do not conform to the job file standard, with one exception which is noted. In order to ensure job file integrity and security, and to prevent accidental restart of jobs interrupted by a system outage, CA Top Secret Batch manipulates the files in the server's reader in several ways during the reader file selection process. Due to the nature of this process, all class 'X' reader files found in the server's virtual reader are automatically purged. Therefore, do not store any files in to the reader for CA Top Secret with a class of 'X'. Note that this purge only affects the server machine's virtual reader; no other class 'X' files in the system are disturbed.

Batch jobs are single-threaded through the server so as not to severely impact the performance of the security system in servicing access verification requests. If you wish, you may submit many jobs at once. They remain in-queue, and a new job starts each time a previous job completes. The order of execution is determined by the sequence of the files in the server's reader queue.

A currently active batch job can be cancelled using the BATCHCAN control option as described in the *Control Options Guide*.

Security Considerations

For access verification purposes, a CA Top Secret ACID and password must be provided in the JCL for each batch job. A pseudo-logon of the user specified is performed during job initiation, and this ACID's security information is used to verify access to the batch facility and to the resources required for the selected batch function. See the heading entitled: "Job Control Statements And Job File Format" below for information on specifying the ACID and password.

Job output, in the form of JCL messages and printed or punched program output, is automatically routed back to the virtual reader of the user submitting the job. Jobs may be submitted from a userid on the same VM system as the server machine, or from a userid on another system via the RSCS network. Networked jobs must contain only standard punched card formats, like those submitted locally, and not contain NETDATA records or DISK DUMP formatted records; therefore, do not use SENDFILE to submit these jobs. A sample SUBTSS EXEC is supplied for submitting jobs over the network.

Due to the sensitive nature of the output of some batch operations, it is important to guarantee that this output is not routed to the wrong user. The originating VM userid of locally submitted jobs is easily identified by CA Top Secret. It must be stressed, though, that such jobs must not be submitted from ID's shared by non-authorized personnel. Otherwise, an unauthorized user has access to program output intended for CA Top Secret security administrators.

CA Top Secret security for batch jobs can be extended to the program level. As a result, in addition to checking for access to the BATCH facility, a security check is also performed for the PROGRAM resource. This allows you to give the administrator access to the TSSUTIL batch programs, for example, without giving him the ability to run any of the others.

Various messages are issued during batch job processing. These not only provide necessary information to the job submitter, but also provide a log of CA Top Secret batch activity in the system.

Job Control Statements and Job File Format

An CA Top Secret batch job file normally is prepared in a standard CMS file. Such a file consists of three sections, each section containing either job control statements or batch program-dependent data.

There are several different job control statements, but all are characterized by the appearance of two slashes ('//') beginning in the first column of the file record. Program-dependent data format is determined by the particular batch program to be executed.

The three sections of a job file are outlined below in the order in which they must appear in the job file:

Job Identification Section

This section identifies the ACID under whose authority the job is to be run and certain other options affecting the entire jobstream. The first statement in a job file must be a job control statement valid for the Job Identification Section. At present, this includes only the TSSJOB statement.

The format of the TSSJOB statement follows:

```
//TSSJOB  ACID=acid,PASSWORD=password
```

where “acid” denotes the CA Top Secret ACID of the user under whose authority the job is to be run, and “password” is that ACID’s password as defined on the local VM system.

Any number of blanks may separate “TSSJOB” from the following arguments.

//DD Control Card

The format of the Audit/Tracking File for Releases 1.2 and above has changed from previous releases. Since auditors are often required to have access to previously created security audit records, a method has been provided to access and report on old format Audit Files. The old format minidisk can be added to the CA Top Secret service machine at any unused virtual address. The release 1.1 file is identified to the TSSUTIL program by including a new control card, //DD, after the //TSSJOB card prior to submitting the job to the CA Top Secret server virtual machine. The syntax of the new card is as follows:

```
//DD AUDIT01,CUU=cuu,DSN=dsn
```

CUU

The address used when defining the old format minidisk to the service machine.

DSN

The data set name used for the old format audit/tracking file.

Note: You cannot produce a merged report of old and new format records using this feature. Inclusion of the //DD statement will provide you with information from a release 1.1 format file, omission of the statement provides you with information from the currently active release 1.2 format file.

Program Execution Section

This section identifies the executable batch program and defines the environment in which it operates. Normally, it consists of multiple job control statements defining input/output devices or data sets, options, etc., terminated by an EXEC statement, which begins program execution. At present, only the EXEC statement is implemented. The EXEC statement format is:

```
//EXEC  PGM=program-name,PARM=exec-params
```

Again, one or more blanks may separate “EXEC” from the arguments, and “program-name” is the name of the CA Top Secret batch utility program to be executed. See the following heading entitled: “Available Batch Utility Programs” for a list of valid program names. Program execution keywords may be supplied via the PARM= option. This is optional, and, if not specified, the comma following the program name must be omitted. If there is only a single execution keyword, and it contains no blanks, commas, or special characters, it may be specified directly after the PARM=; otherwise, it must be enclosed in quotation marks (“) or apostrophes (').

SYSIN Data

All records immediately following the EXEC statement and continuing through end-of-file are treated as program-dependent SYSIN data. Such data include report option specifications for TSSUTIL and batch TSS command scripts for TSSCRIPT. See the associated program descriptions for information on required SYSIN data.

The following is an example of a multiple-report TSSUTIL job stream:

```
//TSSJOB ACID=SECADMIN,PASSWORD=ILUVTSS
//EXEC  PGM=TSSUTIL
REPORT EVENT(VIOL) DATE(TODAY) ACID(SHIFTY) END
REPORT EVENT(INIT) SYSID(SYS1) DATE(-01) END
```

This example shows a program which obtains its input from execution keywords:

```
//TSSJOB ACID=SECADMIN,PASSWORD=ILUVTSS
//EXEC  PGM=TSSRECVR,PARM='DATE(-02)'
```

To submit a job ('TSS INPUT A') in the CA Top Secret server’s (TSSVM) reader enter:

```
SPOOL PUNCH TSSVM CLASS B NOHOLD
PUNCH TSSUTIL INPUT A (NOH
```

Note: In the previous example, the quotes are not actually needed but are included for readability.

Available Batch Utility Programs

CA Top Secret Batch provides several utility programs to aid the Security Administrator or other authorized personnel in system security administration, monitoring system activity, and disaster recovery. Since CA Top Secret security, and Audit/Tracking and Recovery Files may be shared between multiple CPUs and operating systems, these programs were designed to be totally compatible with the corresponding CA Top Secret MVS utilities (CA Top Secret MVS Release 4.3 or above required). Utility functions therefore yield consistent results in a shared CA Top Secret security environment regardless of the system on which they are executed.

The following utility programs are currently available in the CA Top Secret Batch environment:

- TSSAUDIT—Security File changes
- TSSCFILE—Security File reporting
- TSSCHART—Produces block chart of ACID hierarchy
- TSSRECVR—Security File recovery
- TSSCRIPT—Batch TSS command script execution
- TSSUTIL—Audit/Tracking File activity reporting
- TSSXTEND—Security File expansion
- TSSCPREC—List contents of CPF recovery file

TSSAUDIT

TSSAUDIT is a batch utility program that allows the auditor to monitor changes to the CA Top Secret Security File. The type of security information depends upon the control statements selected. Each control statement is discussed in detail following a description of the JCL necessary to execute TSSAUDIT.

TSSAUDIT can be used to perform the following tasks:

- List changes made to the CA Top Secret Security File. A change by a specified ACID or all changes can be listed, a date or range of dates, and a specified string, if desired.
- List Security File information about all ACIDs including attributes and privileges. report on old format (pre-1.2) Audit files.

Note: Due to the large storage requirements of this job, it is strongly recommended that only one request per run be used.

TSSCFILE

TSSCFILE is a batch utility that gives the user the ability to produce customized reports with information extracted from the CA Top Secret Security File. By specifying TSS LIST command(s) as input to the utility, you can select information desired from the CA Top Secret Security File.

The utility parses the output of the TSS LIST command processor and produces a file of fixed format records which, in turn, can be used to generate customized reports using any report writing utility, such as TSSREPT or a customer-written program. TSSCFILE is executed in the same manner as TSSCRIPT, substituting “TSSCFILE” as the program name in the JCL “//EXEC” statement. The rules for coding TSS commands for input to TSSCRIPT described in the preceding paragraphs also apply to TSSCFILE. Please refer to the CA Top Secret *Reporting Guide* for complete information on TSSCFILE and TSSREPT.

TSSCHART

TSSCHART is a batch utility program that builds a tree structure of the full CA Top Secret Security File in memory consisting of control blocks representing zones, divisions, departments, profiles and users. This tree structre is then filtered, depending on user parameters. These parameters, which reside in the SYSIN DD file, are completely in free format and can come from a dataset of any LRECL size. The tree structure is also automatically filtered according to the administrator’s scope. After the tree structure is filtered, TSSCHART “walks through” the tree and uses the Security File to print more detailed information. Please refer to the CA Top Secret Reporting Guide for complete information on TSSCHART.

TSSRECVR

The TSSRECVR utility is an aid to recovery from loss or corruption of the CA Top Secret Security File. During normal system operation, CA Top Secret security maintains a record of all changes to the Security File in the Recovery File. The Recovery File is a perpetual file; that is, change records are added to the file in a wraparound format, and the file must be sufficiently large so as to accommodate all changes that occur between Security File backups.

In the event of loss of the Security File, CA Top Secret is activated using the backup Security File. TSSRECVR may then be used to extract all changes made subsequent to the last Security File backup. TSSRECVR formats selected recovery file change records into a batch TSS command script which may be applied to the backup copy of the Security File using TSSCRIPT (the next heading discusses TSSCRIPT).

TSSRECVR accepts change record selection criteria via the keywords TIME or DATE, which are specified in the EXEC PARM. The correct format follows:

```
EXEC PGM=TSSRECVR,PARM='TIME(hhmm),DATE(yyddd)'
 - or -
EXEC PGM=TSSRECVR,PARM='TIME(hhmm),DATE(-nn)'
```

Where:

- *hhmm*—Is the hour and minute for selecting recovery records. This should be the time of the last Security File backup.
- *yyddd*—Is the earliest date (in Julian format) for selecting recovery records.
- *-nn*—Is the number of previous days for which you wish to retrieve changes from the Recovery File.

To use the contents of the entire Recovery File, specify TIME(0000),DATE(00000).

The following sample CA Top Secret batch job file invokes TSSRECVR to recover all changes made to the CA Top Secret Security File after the last Security File backup at 1:00 AM today.

```
//TSSJOB ACID=SECADMIN,PASSWORD=ILUVTSS
//EXEC   PGM=TSSRECVR,PARM='DATE(-00),TIME(0100)'
```

The resultant TSS command script is returned as a PUN file to the batch job submitter's reader along with the job listing. TSSRECVR abend codes for the VM environment follow:

Abend Code	Reason
2	Printer I/O error
10	Insufficient storage available
11	Recovery File open failure
12	Recovery File
13	Premature end of file on Recovery File
14	I/O error on Recovery File
21	Recovery record too large
22	Bad record type code in recovery record
30	Recovery is active; cannot run
31	ACID not authorized to run TSSRECVR
41	I/O error on punch
1703	Error in execution keywords

Note: If the TSS command contains the keyword TARGET when it is placed in the recovery file on the system it was entered, the TARGET keyword will be commented out and replaced with TARGET(=). This is to prevent duplicate permits on remote nodes when recovery is done on one system. For example:

```
TSS TARGET(=,NODE2) PERMIT(USER1) DSN(ABC.) ACCESS(READ)
```

Will show up in the output from the TSSRECVR utility as:

```
TSS TARGET(=) /*RGET(=,NODE*/ PERMIT(USER1) DSN(ABC.) ACCESS(READ)
```

TSSCRIPT

TSSCRIPT executes a series of TSS commands (a “command script”) in batch mode and returns the output produced by these commands to the job submitter. VM users may realize that a CMS EXEC can be used to execute multiple CP TSS commands automatically; however, doing so might tie up the user’s virtual machine for a significant period of time. Moreover, the only way to capture the output from these commands is to spool the virtual console before invoking the EXEC, resulting in a poorly formatted record of command activity.

The purpose of TSSCRIPT is to provide CA Top Secret users with a method of executing TSS commands. TSSCRIPT is executed in a CA Top Secret batch job. The TSS command script is provided to the program as SYSIN data; EXEC keywords are not supported and are ignored.

Each TSS command must begin with the characters “TSS” in the command script. This may be preceded by one or more blanks, and at least one blank must separate it from the remainder of the command. A record consisting of the characters “TSS” alone is invalid.

Since TSS commands may be longer than an eighty-character punched statement, command continuation is supported. A command may be continued by placing a hyphen ('-') in the last non-blank position in the record. All characters, including blanks, up to but not including the hyphen itself are included in the input to the TSS command processor, and the command is continued starting at the first non-blank character in the following record. If continuation is indicated but no continuation record is found (that is, end-of-file is encountered, the next record is blank or begins begins with “TSS”), an error message is generated, the command is NOT executed, and TSSCRIPT terminates immediately

A command may be continued over several input records however; the total length of the resultant command must not exceed 1270 bytes. By default the TSS command can be entered in any case and will be upper case before it is executed. In some cases, Linux for example, the data is allowed to be in mixed case. In order to support the ability to use mixed case data, TSSCRIPT has two control cards that must being in column 1 and be the only information on the record.

CASE UPPER

All records following this card should be upper case prior to use until the next control card.

CASE MIXED

All records following this card should remain in the entered case until the next control card.

The default is CASE UPPER.

Note: All TSS keywords must be in upper case when entered.

Examples

Examples of commands submitted using TSSCRIPT:

```
TSS ADDTO(USER1) -
    LINUXNAM(Some.Linux.Name)
```

Will get added to USER1 as LINUXNAME(SOME.LINUX.NAME:

```
TSS ADDTO(USER1) -
CASE MIXED
    LINUXNAM(Some.Linux.Name)
```

Will get added to USER1 as LINUXNAM(Some.Linux.Name)

```
TSS ADDTO(USER1)-
CASE MIXED
    linuxnam(Some.Linux.Name)
```

Will get “TSS05501 Unknown Keyword – linuxnam” because the keyword linuxnam must be entered in upper case.

```
TSS ADDTO(USER1)—
  linuxnam(-
CASE MIXED
“Some.Linux.Name”-
)
```

Will get added to USER1 as LINUXNAM(Some.Linux.Name) because the keyword “linuxnam” is before the CASE MIXED and therefore automatically uppercased by the system. Care must be used to not have blank characters in the data portion of the field.

Comments are supported in the input TSS command script. A comment region is defined as one or more input records or portions of input records delimited by the comment initiator and comment terminator strings. The comment initiator is a slash followed by an asterisk (/*) and the comment terminator is the inverse, an asterisk followed by a slash (*/). A comment region may begin and end at any character position of any record of the command script, and may span multiple input records. A comment region essentially is treated as a zero-length field by TSSCRIPT; that is, all characters within and including the comment delimiters are logically omitted from the input record before its use in TSS command construction.

However, each line is still treated as a separate input entity. If a comment is wholly contained in one input record, all characters immediately following the comment terminator are logically concatenated to the characters immediately preceding the comment initiator. In the case of a multiple-record comment, the text preceding the comment initiator on the first record is treated by command construction as one line; text following the comment terminator on the last record of the comment region is also treated as a separate line; any intervening input records are ignored.

Note that all comment removal is done prior to line continuation and command construction, and line continuation is not assumed for multiple-record comments. You must be sure that any required continuation characters fall outside the comment region subsequent to comment removal. For your convenience in debugging and interpreting the results of command script execution, message TSS0540I displays the intermediate text of script input records used as input to TSS command construction after comments have been removed. Nested comments are not supported.

The following example job file illustrates CA Top Secret batch TSS command script execution:

```
//TSSJOB ACID=SECADMIN,PASSWORD=ILUVTSS
//EXEC  PGM=TSSCRIPT
TSS Create(USER2) Type(user) - /* This is a comment */
      Name('John Q. User') -
      Password(educated)  - /* Temp Password */
      Dept(Payroll)
/*TSS Create(USER3) Type(user) - This entire...
      Name('Jane M. Doe')   - command has been...
      Password(FAWN)       - commented out.
      Dept(Woodland)
*/
 TSS Add(USER2) Profile(PayProf1)
 TSS List(Acids) Dept(Payroll) Data(Profile-
    ,XAUTH)
```

TSSUTIL

TSSUTIL provides the security administrator with customized reports detailing security-related activity logged in the CA Top Secret Audit/Tracking File. Reports can be generated for a single CA Top Secret facility or for multiple systems in a shared environment.

Report selection specifications may be entered as SYSIN data or as PARM keywords.

TSSXTEND

The TSSXTEND utility allows you to enlarge or reduce the size of your CA Top Secret Security File, as well as to change your encryption key. TSSXTEND is run in two parts:

- A new larger/smaller Security File is created
- TSSXTEND is use to copy the old Security File into to newly created File.

TSSCPREC

The TSSCPREC utility allows you to list the contents of the CPF recovery file via batch processing.

Chapter 10: Using the NDT Record

This section contains the following topics:

About the NDT Record

The NDT is a reserved or special ACID and global record similar to the Resource Descriptor Table and the Field Descriptor Table. The NDT contains data for assigning PassTickets and Session Keys to applications. The NDT is also used to store LINUXNODE, and CPFNODE records.

You must have MISC1(NDT) authority to maintain data in the NDT.

Define Data to the NDT

To assign a session key to an application, enter:

```
TSS ADDTO(NDT) PSTKAPPL(application) SESSKEY(session-key)
```

PSTKAPPL

Specifies the application assigned a session key for PassTicket processing, and allows one application per command that can be a letter, number, or special character.

Range: Up to 8 characters

SESSKEY

Specifies a hexadecimal "password" that is unique to each application defined by a PSTKAPPL keyword. You must supply a SESSKEY with PSTKAPPL.

Range: Up to 16 characters

Example: defining data to the NDT

This example indicates that the session key for KA180987 is A1B2C3:

```
TSS ADDTO(NDT) PSTKAPPL(KAI80987) SESSKEY(A1B2C3)
```

Define Linux Nodes as NDT Node Elements

To define linux nodes to the CA Top Secret database as NDT node elements enter:

```
TSS ADD|REMOVE|REPLACE(NDT) LINUXNODE(node_name)
                    [IPADDR(ip_address)
                    FACILITY(facility_name)
                    ACTIVE(YES|NO)]
```

LINUXNODE(*node_name*)

Specifies the linux system name.

Maximum length: 246 characters

IPADDR(*ip_address*)

Specifies the IP address for the Linux system.

FACILITY(*facility_name*)

Specifies the facility name to use for system entry validation.

ACTIVE(YES|NO)

Indicates whether the node is active and whether system validation is performed:

YES

The node is active and system validation is performed.

NO

(Default) The node is inactive and system validation is not performed.

Example: defining a linux node

This example adds the new Linux system name LSERVER with IP address 201.212.0.8, using facility LNXPROD for validations:

```
TSS ADD(NDT) LINUXNODE(LSERVER)
          IPADDR(201.212.0.8)
          FACILITY(LNXPROD)
          ACTIVE(YES)
```

Change Values in the NDT

To replace an application, enter:

```
TSS REPLACE(NDT) PSTKAPPL(application)
```

PSTKAPPL

Changes an application that maintains the same session key for PassTicket processing, and allows one application per command that can be a letter, number, or special character.

Range: Up to 8 characters

Example: changing a value in the NDT

This example replaces application KA180987 with application KA180999:

```
TSS REPLACE(NDT) PSTKAPPL(KA180999)
```

Remove Data From the NDT

To remove an application and its accompanying session key, enter:

```
TSS REMOVE(NDT) PSTKAPPL(application)
```

PSTKAPPL

Removes an application and its associated session key used for PassTicket processing, and allows one application per command that can be a letter, number, or special character.

Example: removing data from the NDT

This example removes the application KA180987 and its associated session key:

```
TSS REMOVE(NDT) PSTKAPPL(KAI80987)
```

List NDT Information

To list associated applications only, enter:

```
LIST(NDT)
```

To list subtype records, but not session keys, enter:

```
LIST(NDT) DATA(ALL)
```

To list session keys and associated applications, enter:

```
LIST(NDT) DATA(SESSKEY)
```

To list basic ACID information (such as name sizes and dates) only, enter:

```
LIST(NDT) DATA(NONE)
```

To list all records for the given node type (such as LINUX and CPF), enter:

```
LIST(NDT) DATA(node_type)
```

NDT CPF Node Administrative Commands

CPF node definitions can be defined in the NDT record in the CA Top Secret security file. CPF nodes are grouped by system ID. The system ID entry contains the global system defaults for CPF processing. The sysid and node definitions are administered using the following commands:

```
TSS ADDTO(NDT) CPFSYSID(ssssssss)
           CPFTARGET(tttttttt)
           CPFWAIT(xxx)
           CPFRCVUND(xxx)
           RECVCMDS(xxx)
           CPFOUT(rrrrrrrr)
```

CPFSYSID(*cpflocal*)

Identifies the system name to contain identify the records. This matches the value placed in the CPFLOCAL control option.

CPFTARGET(*|LOCAL|AUTO)

Specifies the default value for the TSS command TARGET keyword.

Default: *

CPFWAIT(YES|NO)

Specifies the default value for the TSS command WAIT keyword.

Default: NO

CPFRCVUND(YES|NO)

Indicates whether the local node can receive commands from undefined nodes.

Default: NO

RCVCMDS(YES|NO)

Specifies that the activity for propagated commands received from remote nodes are logged to a journal file.

Default: YES

RECVDSN(*dsname*)

(Valid on z/OS only.) Specifies a dataset used to hold log data when RECVCMDS(YES) is in effect.

CPFOUT(*rrrrrrrr*)

Specifies the z/VM userid the CPF journal files should be spooled to.

Node specific:

```
TSS ADDTO(NDT) CPFSYSID(ssssssss)
               CPFNODE(nnnnnnnn)
               ACTIVE(xxx)
               RECEIVE(xxx)
               SEND(xxxx)
               GATEWAY(xxx)
               BROADCAST(xxx)
               JOURNAL(xxx)
               JOURNALCLS(x)
```

CPFSYSID(*cpflocal*)

(Required) Identifies the system record to receive this node.

CPFNODE(nnnnnnnn)

(Required) Specifies the CPF node name.

Range: 1 to 8 characters

ACTIVE(YES|NO)

Specifies whether the node is available to send/receive commands and passwords.

Default: YES

RECEIVE(ALL|NONE)

Specifies if the local node can receive commands from that remote node.

Default: ALL

SEND(ALL|CMD|PWD|NONE)

Specifies if the local node can send commands to that node. CMD indicates that only administrative command changes are sent to that node. PWD indicates that only password changes and suspensions are sent to that node.

Default: ALL

GATEWAY(YES|NO)

Specifies a node to act as a CPF gateway or CPF server for another node.

Default: NO

BROADCAST(YES|NO)

Specifies that a node is valid to receive password and command changes that are sent to all nodes via TARGET(*) or the CPFTARGET(*) control option.. If NO is specified changes are only be sent when there are DEFNODES associated with the ACID or a node name is specified with the TARGET keyword.

Default: YES

JOURNAL(YES|NO)

Specifies that the activity for a CPF node is logged to a journal file and the log data is written to a SYSOUT file.

Default: YES

JOURNALDSN(dsname)

(Valid on z/OS only) Specifies a data used as a logging file when JOURNAL(YES) is in effect.

JOURNALCLS(x)

(Valid on z/VM only) Specifies the spool class used for this journal file.

Examples: using NDT CPF commands

This example adds a new system to the NDT:

```
TSS ADD(NDT) CPFSYSID(XA10)
```

This example adds a node to the new system:

```
TSS ADD(NDT) CPFSYSID(XA10) CPFNODE(NODE1)
```

This example removes a node definition:

```
TSS REMOVE(NDT) CPFSYSID(cpfsysid) CPFNODE(nodename)
```

This example lists the a specific CPF node definition:

```
TSS LIST(NDT) CPFNODE(nnnnnnn) CPFSYSID(mmmmmmm)
```

This example lists all nodes for a specific system:

```
TSS LIST(NDT) CPFSYSID(sysname)
```

Chapter 11: Command Propagation Facility

This section contains the following topics:

About CPF

The Command Propagation Facility (CPF) lets sites administer multiple Security Files across VTAM-networked systems by propagating TSS commands, as well as user-initiated changes (such as updated passwords, permissions, maintenance, and suspensions) to all or selected nodes within that network. This process is referred to as distributed security processing because it allows the security administrator to distribute any change in a user's authorization rights and restrictions to any node to which that user is defined.

The Command Propagation Facility provides the security environment with:

- Routing of security administration to all or selected nodes within the z/VM, VSE, and z/OS security network.
- Optional synchronous or asynchronous remote command execution.
- TSS command execution with most CA-Top Secret commands— except MODIFY, LOCK, UNLOCK, HELP, and WHOAMI.
- Automatic update of passwords on all connected systems if changed by the user during logon.
- Propagation of user-initiated suspensions for exceeding password and violation limits.
- Optional Journal Files to log commands transmitted to, and responses received from, remote nodes.
- Collecting asynchronous commands in an optional Recovery File so that they can be retransmitted in case of network outage.

Communication Components

To perform distributed security processing, CA Top Secret relies on CAICCI (Common Communication Interface), a component of CA-CIS. These components are CAIENF (Event Notification Facility) and CAICCI (Common Communication Interface).

CAICCI is a common communications facility that enables CA-Top Secret secured nodes to communicate with one another. It provides the VTAM facilities needed to transmit and receive TSS commands.

Note: The word node, when used in this chapter, refers to the unique identifier (SYSID) that is assigned to that node when it is defined using CAICCI. A node is not the same as the VTAM APPLID, although you can use the same names. See the CA Common Services for VM documentation set for more information about defining nodes.

CPF Architecture

To use CPF, the security administrator must first be aware of all remote CPUs and the ACIDs defined to them. Most Security Files are essentially identical; those defined to one site can be defined to a remote site as well. Security Files that aren't identical can have additional ACIDs, permissions, and/or resource ownership definitions which must be taken into account. This can become a difficult task for the security administrator responsible for maintaining user information. If this is the case, you may want to consider using automatic propagation based on default nodes (DEFNODES) for ACIDs.

Implicit and Explicit Targeting

CPF lets you automatically synchronize Security Files on multiple nodes through the propagation of TSS commands as well as user-initiated changes— such as suspension and password changes. Security administration propagation can be **implicit** (by using the CPF control options to set system-wide propagation rules) or **explicit** (by using the CPF command keywords to set propagation rules on a command-by-command basis).

The designated CPF control option values specified in each Parameter File determine the implicit target nodes to be used when a command is issued from that particular node. For example, if the CPF control options for NODEA identify implicit target nodes of NODEB, NODEC, and NODED, whenever a command is issued from NODEA that command is automatically sent to NODEB, NODEC, and NODED.

By using the CPF command keywords, a security administrator can override the targets designated by the control options. For example, even though the control options for NODEA identify implicit targets of NODEB, NODEC, and NODED, the security administrator can use the TARGET keyword to indicate that a particular command should only be propagated to NODEB. For more details about the TARGET keyword, see CA-Top Secret Command Keywords Used With CPF, later in this chapter.

Synchronous and Asynchronous Processing

In addition to designating target nodes for a command, you can also indicate how that command is processed. By using the appropriate control options and command keywords, you can process on a synchronous or asynchronous basis as described below.

Synchronous

Allows the TSS command to execute simultaneously throughout the network. CA-Top Secret waits for the command response to return from the remote node before continuing.

Asynchronous

Allows the TSS command to execute on a selective basis throughout the network. This means that CA-Top Secret does not have to wait for a response from each node to resume processing. All commands transmitted through asynchronous processing are retained in a CPF Recovery File.

These options are independent and can be used separately or together. The synchronous or asynchronous processing of commands and the specific targeted nodes are initially determined by control option settings. Later, these values can be changed using the appropriate TSS command keywords. For more details, see CA-Top Secret Command Keywords Used With CPF, later in this chapter.

Defining CPF nodes

CPF nodes are defined as CPFNODE elements on the NDT system record in the CA-Top Secret security file. CPF nodes are grouped on the NDT record by system id, to allow multiple CA-Top Secret systems use separate CPF node definitions. For downward compatibility, this release of CA-Top Secret, will also support defining CPF nodes using the CPFNODES control option. Once a CPF node is defined to the NDT record, a control option definition for the same node name is ignored.

CPF node definitions in the NDT record can be created or modified at any time after CA-Top Secret starts up. The START & REFRESH commands described later in this chapter are used to activate newly defined CPF nodes, or to apply new processing options to existing active CPF nodes. When refreshing an individual node, CPF processing for other nodes is not interrupted.

The following example shows how to set up the NDT CPFNODE definitions for system SYS1, to allow sending and receiving CPF commands and passwords to remote system SYS2:

```
TSS ADD(NDT) CPFSYSID(SYS1) RECVCMDS(YES) CPFTARGET(LOCAL)

TSS ADD(NDT) CPFSYSID(SYS1) CPFNODE(SYS2) JOURNAL(YES) RECEIVE(ALL)  SEND(ALL)
```

Note: If all CPF related control options are migrated from the control options parameter file to the NDT, control option CPF(ON) or CPF(OFF) is required in the parameters file for CPF to initialize using the NDT based definition. If neither is specified, CPF will not initialize and cannot be activated unless CA-Top Secret is recycled. CPF(ON) activates CPF as soon as CA-Top Secret is started up. CPF(OFF) will start CA-Top Secret with CPF inactive, however, CPF may be later activated via a TSS MODIFY(CPF(ON)) command.

Note: Adding a new CPF node to an active CA-Top Secret system, also requires adding a NODE definition to CAICCI.

Administrative Authority

In all cases, administrative authority and scope of the security administrator is verified at the sending and target nodes before the command is successfully applied to the targeted Security Files. This is an important level of control over remote administration. To enter TSS commands with a targeted destination, you need MISC2(TARGET) authority.

In addition to propagating changes, CPF allows the security administrator to view the contents of Security Files in remote nodes. This viewing is completely secure since scope is verified at both locations, allowing the administrator or auditor to review the security information for which he or she is responsible at all nodes in the CPF domain. CPF also encrypts the data that it sends between nodes.

Propagated administration commands execute on the remote system using the authority and scope of the user as defined on the remote system, and not from the originating system. For example, if a user who is defined as an SCA on one system propagates a TSS command to a system where that user is defined only as a USER, then the command is limited to the USER authority.

User initiated (versus administrator initiated) password changes propagated through CPF cause the user's password to change at each node where the change is sent, provided that the user's existing passwords are the same. The password will not change at nodes where the existing passwords aren't identical or aren't synchronized. This prevents one user from changing the password of an identically named ACID on another node that is used by another person.

CPF-Related Control Options

CA-Top Secret supplies control options that govern the use of CPF and enable distributed security to be maintained efficiently. At least one of the CPF-related control options described below **must** be entered at CA-Top Secret startup to use the Command Propagation Facility. If it is not, CPF cannot be activated until the next CA-Top Secret startup, and no CPF control options are accepted by CA-Top Secret until that time. Once you have designated control options, your TSS commands automatically propagate to the default nodes.

The following control options tailor the environment for the Command Propagation Facility:

CPF(ON|OFF|KILL)

Indicates whether CPF should be activated at CA-Top Secret startup (ON|OFF) and lets you temporarily terminate the CPF subtask (KILL) without bringing down all of CA-Top Secret.

ON—TSS commands are transmitted by this node or received from other nodes.

Note: If CPF(ON) has been specified, but CCI is not available or not fully initialized, CPF status is displayed as CPF(INIT). While CPF is in this status, commands are not propagated via CPF and are not logged to the CPF Recovery File. Once CCI completes its initialization, CPF status will display as CPF(ON), and command propagation and logging will take place.

OFF—No TSS commands can be transmitted.

KILL—Issued with a TSS MODIFY command to temporarily terminate the CPF subtask and automatically take a dump. The subtask can then be reattached by specifying TSS MODIFY(CPF(ON)).

CPFNODES(node1,node2Ä(S)|(R)|(C)|(P)|(GW)|(NB)|,...)

Identifies the remote CA-Top Secret nodes from and/or to which CPF can propagate commands.

(S)—Indicates that the local node can only send commands to the designated remote node.

(R)—Indicates that the local node can only receive commands from the designated remote node.

(C)—Specifies that only administrative command changes and DUF updates are sent to a node.

(P)—Specifies that only password changes and suspensions are sent to a node.

(GW)—Allows a CPF node to act as a CPF gateway or CPF server for another CPF node.

(NB)—Indicates that the node is a no-broadcast node; used when CPFTARGET(LOCAL) is the default.

Note: User-initiated changes (such as updated passwords or suspension due to access violations) or duf updates are propagated to those nodes identified by the CPFNODES control option.

CPFRCVUND(YES|NO)

Indicates whether the local node will receive commands issued from a remote node that hasn't been defined to the CPFNODES list. The default is NO— the local node will not receive commands from undefined remote nodes.

CPFWAIT(YES|NO)

Sets a default value for the TSS command WAIT keyword.

If CPFWAIT is omitted, CA-Top Secret chooses a default of YES. This means that commands are processed on a synchronous basis, requiring the user to wait for the commands to complete on all specified nodes before the local command completes.

If NO is selected, processing occurs asynchronously.

Regardless of whether you select YES or NO, the CPFWAIT control option can be overridden by the WAIT value on the individual TSS command.

CPFTARGET(AUTO|*|LOCAL)

Sets a default value for the TSS command TARGET keyword.

The security administrator can select one of three options.

AUTO—Indicates that, if a target node is not explicitly identified on a command, that command will automatically propagate to those nodes identified by the ACID's DEFNODES. A more complete discussion of the connection between CPFTARGET and DEFNODES is included in the *Control Options Guide*.

asterisk (*)—Indicates all nodes defined as send-only or send/receive in the CPFNODES control option. Nodes defined as receive-only are not included.

LOCAL—Indicates a particular local node.

CPF related MODIFY commands

CPFNODE(nodename=STOP)

This command stops all new activity for the node and the node will show up with status of STOPPED. Commands currently queued up for transmission are processed but no new commands will get queued.

CPFNODE(nodename=START)

This command will activate a previously stopped node, or install and activate a new NDT defined node.

CPFNODE(nodename=REFRESH)

This command will modify the node attributes based on current NDT definitions. New attributes will only affect commands and password changes not yet queued to the node. Recovery file records and in-core queue entries will still be processed based on the attributes in effect prior to the refresh.

Note: Activating a new NDT defined node using the START or REFRESH commands, also requires adding a NODE definition to CAICCI.

CA Top Secret Command Keywords Used With CPF

The TSS command functions that can be used with CPF are authorization commands such as ADDTO, PERMIT, REMOVE, and CREATE as well as, WHOHAS, LIST, and WHOOWNS.

The CPF keywords that can be used with these functions are:

TARGET(node1,node2...)

Identifies each node to which a command can be propagated.

TARGET(*)

Transmits the command to the local node and to all nodes defined in the CPFNODES control option.

TARGET(=)

Restricts command execution to the local node only. The TARGET(=) keyword overrides the CPFTARGET(LOCAL) control option.

TARGET(n...n*)

Transmits all commands to nodes whose names begin with the indicated string. The string can range from one to seven characters.

TARGET(SELECT)

Propagates commands to all DEFNODES for a user, including nodes marked as no-broadcast nodes.

DEFNODES(node1,node2,...)

Defines default remote nodes for use in the event that a security administrator does not specify a TARGET keyword on a command. DEFNODES only applies if the CPFTARGET control option has been set to AUTO.

WAIT(Yes|No)

Sets the processing mode for the command being issued. WAIT(YES) selects synchronous processing. WAIT(NO) selects asynchronous processing. The WAIT keyword overrides the CPFWAIT control option setting.

The example below displays the users on CPU1, CPU2, and the local node that have access to payroll data.

```
TSS WHOHAS DSNAME(PAYROLL.) TARGET(CPU1,CPU2,=) WAIT(Y)
```

The next example grants ownership of the SYS1 data set prefix to DEPT01 on all remote R-prefixed nodes identified by the CPFNODES control option.

```
TSS ADDTO(DEPT01) DSNAME(SYS1.) TARGET(R*) WAIT(Y)
```

The final example designates the ALT, BOS, and CIN nodes as the default routing nodes for USER01.

```
TSS ADDTO(USER01) DEFNODES(ALT, BOS, CIN)
```

Using DEFNODES With a TSS Command

Although asterisk (*) and LOCAL are the more commonly used values with CPFTARGET, when using DEFNODES with a TSS command, AUTO is required.

DEFNODES only applies under two conditions:

- If CPFTARGET is set to AUTO and
- No TARGET keyword is specified on the command.

If these two conditions are met, CA-Top Secret automatically retrieves the DEFNODES normally associated with the **targeted ACID**. The only time the DEFNODES keyword is actually supplied in a command is when the ACID's DEFNODES are being designated (on the initial TSS CREATE or later through a TSS ADDTO) or updated. DEFNODES administration is discussed in the next section.

Note: If CPFTARGET is set to AUTO and no DEFNODES are specified, routing is done to all the nodes with send abilities **only** when a user changes his own password or is suspended because of an invalid password or because he has been inactive for too long.

If the command being issued is an ADD/REMOVE, ADMIN/DEADMIN, DELETE, MOVE, PERMIT/REVOKE, RENAME, or REPLACE, the destination nodes are taken from the DEFNODES of the targeted ACID.

Administering an ACID's DEFNODES

DEFNODES can be assigned to an ACID in one of three ways:

- Using TSS CREATE.

  If you are using TSS CREATE, you need to specify the DEFNODES keyword. For example, the command shown below creates an ACID of USER01 for John Smith and designates NODEA, NODEB, and NODEC as his DEFNODES.

  ```
  TSS CREATE(USER01) NAME('John Smith') TYPE(USER)
        PASSWORD(shsh,30,exp) DEFNODES(NODEA,NODEB,NODEC)
  ```

- Using TSS CREATE with a model ACID.

  If you are using a model ACID, that ACID must have DEFNODES to be transferred to the new ACID. For example, the command shown below uses the ACID created in the previous example as the basis for Sam Jone's ACID, USER02. Since USER01 has DEFNODES, these same DEFNODES are being transferred to USER02.

  ```
  TSS CREATE(USER02) USING(USER01) Name('Sam Jones')
  ```

  Note: Model ACIDs can only be used as a basis for creating other ACIDs of the same TYPE. For example, you cannot use a model ACID whose TYPE is USER to create another ACID whose TYPE is DCA.

- Using TSS ADDTO with an existing ACID.

  You can add DEFNODES to an existing ACID. For example, the command shown below indicates that ACID01 now has DEFNODES of NODEB and NODEC.

  ```
  TSS ADDTO(ACID01) DEFNODES(NODEB,NODEC)
  ```

Note: If a TSS CREATE is issued **without** indicating the DEFNODES keyword, or by using a model ACID that does not have DEFNODES, no DEFNODES are supplied.

In addition to TSS ADDTO, you can also use TSS REMOVE and TSS REPLACE to update an ACID's existing DEFNODES definitions.

TSS REMOVE is used to delete one or more entries. For example, the command shown below removes NODEB from USER02's DEFNODES list while leaving NODEA and NODEC.

```
TSS REMOVE(USER02) DEFNODES(NODEB)
```

TSS REPLACE deletes an ACID's existing DEFNODES definitions **in their entirety** and replaces them with a completely new list. For example, the command shown below completely removes the current DEFNODES from USER02 and replaces them with NODEF, NODEG, and NODEH.

```
TSS REPLACE(USER02) DEFNODE(NODEF,NODEG,NODEH)
```

Note: A maximum of five DEFNODES can be added, removed, or replaced in a single TSS command.

What Happens When a Command is Issued

To get a better understanding of how the CPF control options and CPF command keywords work together, consider the consequences of the following sample commands:

```
TSS ADDTO(USER01) DSNAME(ABC123)
```

Since no TARGET was specified, CA-Top Secret looks to the CPFTARGET control option setting to determine the designated default routing procedure:

- If CPFTARGET is set to AUTO, the command will propagate to those nodes identified by the ACID's DEFNODES.
- If CPFTARGET is set to LOCAL, the command will only execute on the local node.

- If CPFTARGET is set to *, the command will propagate to all nodes identified by the CPFNODES control option, except for nodes identified as no-broadcast nodes.
- If CPFTARGET hasn't been activated, the command will only execute on the local node.

```
TSS ADDTO(USER01) DSNAME(ABC123) TARGET(*)
```

In this case, USER01 is being granted ownership of the ABC123 data set. Since a TARGET of * was specified, this command is propagated to all nodes identified by the CPFNODES control option.

```
TSS ADDTO(USER01) DSNAME(ABC123) TARGET(=)
```

In this case, the same command will only be executed on the local node (as specified by the =).

```
TSS ADDTO(USER01) DSNAME(ABC123) TARGET(NODEA,NODEB)
```

Here the command is propagated to NODEA and NODEB regardless of the values specified by the CFPNODES or CPFTARGET control options and any DEFNODES USER01 may have. An explicit TARGET will override the defaults.

```
TSS ADDTO(USER01) DSNAME(ABC123) TARGET(*,SELECT)
```

This command is propagated to all broadcast nodes and any no-broadcast nodes that are identified as DEFNODES for USER01.

CPF Recovery File

The CPF Recovery File is a BDAM disk file used by CPF to save transmitted commands until a response to those commands has been received from remote machines. Only commands selecting or defaulting to WAIT(NO) are saved for retransmission. Commands targeted only for the local machine are not saved.

When a TSS command with WAIT(NO) is entered, CPF saves a command image on the CPF Recovery File before transmitting it over the link. When a response from the remote node is returned, CPF deletes the command from the file. If the response is not received due to link failure, system down, etc., CPF will scan the Recovery File at the resumption of service and select from it all commands that were sent out but not responded to, and retransmit those commands. When a response is eventually received, the command(s) will be deleted from the file.

CPF performs a scan of the Recovery File at the following times:

- At CA Top Secret initialization.
- When contact with a remote machine is reestablished after having been lost.

Before you can use the Recovery File, it must be formatted through TSSMAINT. The CPF Recovery File must be defined to the CA Top Secret server virtual machine at address 600.

Warning: The CPF Recovery file cannot be shared across multiple systems.

If the CPF Recovery File is not defined, command routing through CPF can still occur, but there will be no retransmission of unresponded commands. See the *Getting Started* for more information about the CPF Recovery File.

If the CPF Recovery File becomes temporarily filled, a message is written to the system operator's console each time CPF wants to write a message to the file but cannot. The CPF operation continues but, in case of failure, the unwritten command cannot be recovered.

CPF Journal Files

CPF uses Journal Files to provide an historical record of the command traffic to and from CA Top Secret. An individual Journal File is a print file which can be printed offline or viewed online. CPF will optionally allocate one Journal File for each remote node defined to it through the CPFNODES control option plus one Journal File for all incoming traffic. (Remember that CPFNODES specifies the places that CPF can send to but does not affect from where it can receive.)

When CPF transmits a command to a remote destination, it records the command image on the Journal File for that node and associates an ID number with that command. When a response is received from the remote node, CPF journals the response and the ID number so that the response can be matched to the command that prompted it. Similarly, when a command is received from a remote machine, CPF journals the command, the ID number, and the node name that sent the command.

When the response is sent back, it is journalled along with the ID and remote destination name. Thus, by examining the appropriate Journal File, an auditor can see exactly what came in, what went out, and the results of the action taken. Unlike the CPF Recovery File, the CPF Journal Files log commands regardless of whether WAIT(YES) or WAIT(NO) was specified.

Recovery and Accountability

CA-Top Secret provides recovery processing services for the Security File through the TSSRECVR utility. (No other files can be recovered using this utility.)

TSSRECVR Utility

When you run TSSRECVR (the CA-Top Secret utility that generates TSS commands to recreate the Security File), the TARGET is always local no matter what the original TARGET destination.

TSS Command Execution

Anyone with MISC2(TARGET) authority can enter a TSS command with a targeted destination. The command is executed at the remote machine under the authority the issuing ACID has on the remote node (regardless of his authority on the local node).

For example, ACID(HARRY) is defined as an SCA on his local machine, but on REMOTEB ACID(HARRY) is only a USER. Any command HARRY sends to REMOTEB is executed with his authority on REMOTEB— as a user.

Note: If the security administrator issuing the command does not exist on the remote node to which the command is being propagated, you will receive the following message:

```
TSS0324E ADMINISTRATOR'S ACID DOES NOT EXIST ON TARGET NODE
```

To avoid this error, you should ensure that the administrating ACID is defined to the remote node before any commands are issued to that node.

Security Files Among Networked Machines

As stated earlier, each command routed by CPF is executed at the remote machine as though it originated there.

Resources, ACIDs, and so forth, may or may not exist on one or more remote node machines. Commands that work at one remote may fail at another; and commands executed at a remote node may not have their intended effect.

Using the previous example, if the MSCA gave ownership of DSNAME(SYS1) to ACID(HARRY) at his local node (where he is designated as an SCA) that command, if routed to REMOTEB through CPF, can give the same ownership to USER(HARRY).

System Entry Validation and Password Propagation

When a user is required to update his password as part of system entry validation, CA-Top Secret notifies all other CPF connected nodes of that change. If a matching ACID is found on a remote node, CA-Top Secret first compares the password of the "remote" ACID with the old password of the "local" ACID to verify that they aren't two different ACIDs with the same ACID name. If the passwords match, the password on the remote node is updated.

For example, USER01 signs on to TSO on Node A. His current password of "remember" has expired and he changes it to "always". Through CPF, that change is sent to Node B, a remote node. When CPF finds a USER01 in the Security File of Node B, it asks "Is the current password remember?".

- If the answer is yes, the password of USER01 on Node B is also updated to "always".
- If the answer is no, CPF assumes that the USER01 on Node B is not the same person as the USER01 on Node A and leaves the Node B password unchanged.

When CPF detects that the passwords do not match, the following message is sent to the CPF spool data set on the remote node:

```
TSS0422E PASSWORD VERIFICATION FAILED ON REMOTE NODE
```

Installation Exit

The installation exit routine can be called for CPF transmission on both the sending and receiving side, and may block the command at either point. The exit can also make some changes to the command text. (Refer to the installation exit code for instructions).

TSSCPR Utility

The TSSCPR utility is designed to show the submitting administrator which TSS commands are still pending in the optional CPF Recovery File. The TSSCPR report identifies the origin of the command, the targetted node, a transaction identification number, and the command itself. Passwords are always replaced by a “?”.

Chapter 12: Recovering from Security File Loss

This section contains the following topics:

Security File Recovery Strategy

You should familiarize yourself with these steps and, in consultation with your Systems Programming staff, use this document as a guide to developing your own particular operating procedures and recovery strategy.

Implementation of a recovery strategy consists of the following parts:

- Designating Backup-Responsible System
- Managing Data Sets and Tape Backups
- Selecting Host System(s) for Recovery
- Reestablishing the System After Loss
- Reconstructing the Security File
- Reinstating Normal Operation

While each of these steps may vary according to your particular needs or system configuration, all are essential components of an effective backup and recovery implementation.

Designating Backup-Responsible System

During installation of CA Top Secret VM, you are asked to define certain database sharing options. Specifically, you must indicate whether the Security Data Base is shared among multiple CA Top Secret systems and, if so, whether the system you are installing is responsible for performing the DASD-to-DASD backups of the Security File. This daily disk-to-disk copy of the Security File to the Backup File is an integral part of CA Top Secret's built-in recovery features; therefore, you should ensure that one, and only one, system is operating at all times with the BACKUP(hhmm) control option set to an appropriate time, while all others in a shared environment have BACKUP(OFF). In a non-shared environment, be sure that BACKUP is enabled in the CA-Top Secret VM startup keywords.

For the purpose of obtaining this daily backup in a shared environment, it does not matter which system (nor type of host operating system - VM, MVS, or VSE) is selected to perform backups. Your choice should be based on your systems' operating schedules, the availability and requirements of automated operations tools of which you might wish to make use to trigger the tape backup step (below), or your operations staff's schedule and responsiveness to different systems during off-hours.

Note also that neither tape backups nor Security File reconstruction has to be done on the system that performs the backups.

Managing Data Sets and Tape Backups

Throughout nearly every working day, changes are being made to the records in the Security File, whether explicitly by security administrators using the CP command, implicitly by users' changing their passwords, or automatically by CA Top Secret as through suspension or expiration of ACIDS. Therefore, proper backup and recoverability is a perpetual concern, because at no time can a backup CA Top Secret Security File be guaranteed to be truly current. Fortunately, each Security File change is also written as a change record to the CA Top Secret Recovery File. By combining the change records with the most current CA Top Secret Backup File, you can easily recreate your Security File with no loss of information.

In order for this recovery mechanism to function, both the Recovery File itself and a good backup obviously must be available in the absence of the Security File. This means that the keys to recoverability are placement of data sets and timing of disk-to-tape backups; both of these must be addressed in your recovery planning. In doing so, you should consider two possible recovery scenarios: the first is loss of only the Security File, and the second, loss of the entire Security Data Base.

The first scenario is partially an engineered situation; that is, it is the easiest from which to recover and the ideal to which you should strive in allocating your CA-Top Secret security data sets. Since catastrophic loss of access to data is most commonly associated with a single DASD unit or actuator (whether due to DASD hardware or media failure, accidental scratch, loss of a path to the device, or some other cause), you naturally want to ensure that such a failure will not take out both the Security file and the components needed to rebuild it. Placing the Backup and Recovery files on a separate physical DASD unit (ideally on a string attached to a different storage director or, if possible, a different channel) from that of the Security File reduces the likelihood of such a multiple loss. In this case, if access to the Security File is lost, you can bring CA Top Secret up on the Backup File, apply the changes from the Recovery File that is already online, and quickly resume full security system operation while you work to resolve the problem.

You must also prepare for the second scenario, in which you do not have immediate access to a valid Backup File and/or Recovery File, or the Backup File is not usable. In such a case, you need to restore one or both from tape before you can attempt to restart CA Top Secret. Your recovery implementation plans need to include specific procedures for the backup and offsite storage of all CA Top Secret data sets. The best time to back up these data sets is immediately following CA Top Secret's automatic disk-to-disk daily backup. CA Top Secret quiesces all updates to the Security File for the duration of the automatic backup to prevent propagation of partial updates and ensures that the backup copy is usable. This backup copy is the preferred data set to restore from tape should the need arise, although it is recommended that you back up the Security File as well, just to be safe.

Whatever method or product you use to back up these files, you must be sure that all the necessary tools are available when the time comes to restore it. For instance, if you use an MVS-based utility to produce the backups and need to restore them on a VM system, be sure that you have a standalone version of the utility for use in a virtual machine.

It should also be noted that, if the current online Recovery File is unavailable, you probably are not able to fully reproduce the lost Security file. Assuming that you have access to the most recent backup file, or to a backup of the Recovery File, your data should be no more than twenty-four hours out of date.

Selecting Host System(s) for Recovery

If your environment consists of two or more CA Top Secret systems sharing the Security Data Base, you need to choose the system or systems from which you will perform Security File recovery tasks. The recovery tasks are the actual steps required to activate CA Top Secret in limited backup mode, restore any files if necessary, and reconstruct the Security File. While it may not be appropriate or necessary to implement recovery procedures for all systems sharing the same Security File, you should, at the very least, choose a system and method which can function independently, without relying heavily on other systems to complete the tasks.

For instance, your plan should not include submission of batch jobs across CPUs to an NJE system that does not accept jobs unless CA Top Secret is available. Likewise, if your implementation requires that certain tasks be performed by a guest MVS system under VM, be sure that the MVS system does function properly without relying on any special resources or privileges normally granted by CA Top Secret running on the host VM system; if it does, you should be sure that you can also recover the file using available utilities and CA Top Secret VM on the host VM system.

The remainder of this section discusses recovery implementation using a VM system and CA Top Secret VM. For further details on implementing recovery using CA Top Secret MVS or VSE, see the CA Top Secret MVS or VSE documentation.

Reestablishing the System After Loss

The first step to reestablishing normal operation after the CA Top Secret Security File has become unusable is to get CA Top Secret running again on a Backup Security File. In most cases it is desirable to develop a procedure that lets this task to be performed by data center operations personnel, after obtaining proper authorization, during off-hours so that down time will be minimized.

Note: You should activate only one CA Top Secret Security system for recovery purposes in a shared security environment unless you need to operate using the Backup File for an extended period of time.

If the Backup file is intact and accessible, then this can be accomplished quickly and easily using the special AUTOLOG/XAUTOLOG procedure below.

Note: Use the XAUTOLOG command when running in VM/XA or VM/ESA mode.

If the Backup file is not intact, you need to restore the most recent copy of the Backup file to disk. If your DOWN option is WAIT or FAIL, then you may have some difficulty with this. If so, issue the command and execute the appropriate commands to disable additional terminal logons:

```
TSS MODIFY(DOWN(VN)) (or VB)
```

The procedure for restoring the Backup File and/or minidisk will vary depending on your method of backup. Typically, this involves using DDR or a the standalone restore utility supplied with your DASD utility software. You may also wish to include in your procedure the creation of a tape copy of the Backup File before bringing up the server, to insure against incorrect application of changes later on.

Now that you have ensured that the Backup File is online and available, CA Top Secret VM may be initialized in backup mode, as follows:

- Ensure that the server is not logged on. Presumably, it has logged itself off as a result of the failure. If not, use TSS MODIFY(SHUTDOWN) to deactivate security. The server should now log itself off.
- Reactivate security in backup mode, using a special form of the XAUTOLOG command that passes variable data to the server machine. You supply as variable data the keyword BACKUP, which the server recognizes as a command to invoke backup mode. Ask your VM Systems Programmer how the PASSWORDS_ON_CMD suboption AUTOLOG is set. If AUTOLOG YES is set you should use the command:

  ```
  XAUTOLOG ServerID BACKUP
  ```

  where ServerID is the VM userid of the CA Top Secret VM server (typically TSSVM). If AUTOLOG NO is set the command format is:

  XAUTOLOG ServerID password BACKUP

  where ServerID is the VM userid of the CA Top Secret VM server (typically TSSVM), and password is the server's CP directory password.

If AUTOLOG NO, something -- such as 'XXX' -- must be typed in place of the password to satisfy VM's command parser but it is ignored. However, if CP has been IPLed since the last time security was active, then CA Top Secret's CP component does not know the ID of the server, so the correct directory password must be supplied.

If you wish to use XAUTOLOG, you need to have the logical-line-end character precede the console_input_ data in the command, as follows:

```
XAUTOLOG Server ID #BACKUP
```

If, however, the issuer of the XAUTOLOG command is using CMS, then CMS will truncate the XAUTOLOG command and the logical-line-end character and not pass the parm BACKUP to the virtual machine. To prevent CMS from interpreting the logical-line-end character in this manner one should either precede the character with the logical-escape character, or preface the entire command with #CP as follows:

```
XAUTOLOG Serverid "#BACKUP

or

#CP XAUTOLOG Serverid #BACKUP
```

where '"' is the logical-escape-character and '#' is the logical-line-end-character (see the IBM CP command and Utility Reference).

In addition, clients who have set up their CA Top Secret server to ipl CMS, and then later to issue 'IPL 100' in its profile exec, should note that the XAUTOLOG command described above will cause the server to come up on the Backup File. The only way to achieve this result is to override the IPL statement (and the (MACHINE statement, if MACH ESA is entered) in the server's directory entry, as follows:

```
Non-CMS issuer:
XAUTOLOG ServerID IPL 100 #BACKUP
CMS issuer:
XAUTOLOG ServerID IPL 100 "#BACKUP
or
#CP XAUTOLOG ServerID IPL 100 #BACKUP
```

The server should now IPL normally, with the addition of the following actions upon recognizing the BACKUP keyword:

The 200 (Security File) minidisk is detached so that the real device may be varied offline, if necessary, for problem determination or hardware service.

The Backup File data set name and virtual address (500) is used in place of the Security File.

Recovery file logging and automatic backup are disabled regardless of the control options in the Parameter File.

Upon successful initialization, the following message appears on the VM operator's console:

TSS0030I **WARNING** TSS/VM Running on BACKUP File - DSN: dsn

This message confirms successful backup mode initialization and reminds you that the system is running on the Backup File and should be carefully managed until normal operation is restored. If necessary, it should now be possible to re-enable system logons (if previously disabled) and attain a reasonably normal level of system operation.

Reconstructing the Security File

Now that CA Top Secret is functioning once again (hopefully on a reasonably up-to-date Backup File), the task of recovering recent Security File changes may be undertaken at a convenient time (or when the necessary personnel are available). Note that this should be done as soon as possible, since recent security definitions and password changes may have been temporarily negated.

The first step is to extract the applicable change records from the Recovery File. Of course, if the recovery file has been damaged or lost, the changes must be repeated manually. Adequate hardcopy records of security administration should be kept on a day-to-day basis to accommodate this situation.

Provided the Recovery File is intact, you may now use CA Top Secret's TSSRECVR utility to produce a TSS command script of all administration and implicit Security File changes incurred since the last automatic backup (or since the now-online Backup file was produced). Details on the operation of TSSRECVR and the CA Top Secret VM batch facility under which it executes may be found in the Batch Operations section in this guide. See that section for guidance in preparing and submitting the necessary Job Control Language; for purposes of illustration we will discuss here only the control statements and keywords required by the utilities.

Assuming the last backup was performed at 01:00 on the date of the failure, the following EXEC statement would be used:

```
//EXEC PGM=TSSRECVR,PARM='DATE(-00),TIME(0100)'
```

Successful execution of TSSRECVR produces two files in the submitter's virtual reader: a PRT (print, or listing) file and a PUN (punch) file. The PRT file is the SYSOUT listing of the utility's execution. It contains a log of the supplied execution keywords, a description of the age and contents of the Recovery File, the number of change records of various types extracted, and the overall condition code of the run.

You should inspect this file to verify that the expected results were obtained from your request.

The PUN file contains the resultant TSS command script. This script consists of actual Security Administrators' commands as well as simulated TSS command equivalents of automatic and user-generated changes (such as suspensions and password changes). Preceding each command is a comment record identifying the nature and source of the change and the date and time of its occurrence. Receive this file onto disk and, if desired, you may review and/or modify it in any way you see fit.

To apply the changes to the current Backup Security File, the TSS command script must be submitted for batch execution as input to the TSSCRIPT utility program.

Using a text editor (for instance, the VM/System Product Editor, XEDIT), add appropriate Job Control Language to the command script as described in the Batch Operations section of this guide and submit the file for execution. When the script completes, a listing file is returned to the submitter's reader in which each command is reproduced along with its result, followed by an overall completion summary. Review this listing for errors and, if necessary, correct and resubmit any portions which did not complete successfully.

Reinstating Normal Operation

The Backup File, on which your CA Top Secret system is running, should now be an exact duplicate of the Security File that was lost. Before returning CA Top Secret to its normal operational state, the Security File must be recreated. Since the Backup File already contains all of the correct security definitions, this is accomplished simply by a reversal the original backup process. When operating in backup mode, CA Top Secret responds to an explicit backup command by doing just this -- copying the Backup File onto the Security File -- so great caution must be taken in using backup mode and, especially, the TSS MODIFY(BACKUP) command while in backup mode.

Below are the specific steps that should be taken to recreate the Security File:

1. Ensure that CA Top Secret is operating in Backup mode.
2. If the Security File was permanently lost or damaged, reallocate and format it at this time. Remember that if the Security file is to be allocated on a real z/OS or VSE-formatted volume, this MUST be done using the CA-Top Secret z/OS or VSE TSSMAINT utility; if on a VM-only minidisk, re-execute CA Top Secret VM Installation Task 6, Define Security Data Base Files, and elect to format only the Security File. Be sure to specify the same keywords originally used to create your Security File.
3. Ensure that the DASD volume on which the Security File resides is online to VM and, if defined as a minidisk to the server, attached to the system. The server automatically LINKs the minidisk if necessary when the backup operation begins. If the Security File is on a full-pack OS volume which is normally dedicated to the server on the VM system, you must verify that it is attached to the server as virtual address 200. Consult your Systems Programmer if you need assistance with this procedure.
4. Issue the command TSS MODIFY(BACKUP). This triggers an immediate backup that, in backup mode, is performed in the reverse -- building a new Security File as an image of the current, newly updated backup file. When the backup completes, you will have successfully reconstructed your security system. However, CA Top Secret is still running in backup mode, using the Backup File as its active database and with change recording inactive. It is essential to reset CA Top Secret to normal operation as soon as possible after the Security File has been recreated, as described in the next two steps.
5. Deactivate CA Top Secret by issuing the TSS MODIFY(SHUTDOWN) command. The server terminates backup mode security operation and logs itself off. At this time, you should also shut down any other CA Top Secret system (VM or MVS) that is also running in backup mode.

   Warning: Running multiple CA Top Secret systems simultaneously on separate Security Files but sharing the same Audit and Recovery files circumvents CA Top Secret's locking mechanisms and can result in data base corruption. As noted earlier, you should restrict backup mode operations to a single system, if possible, to avoid accidents.

6. Re-activate CA Top Secret using a normal AUTOLOG command without the BACKUP keyword. CA Top Secret initializes as usual, using the new Security File.
7. If you are sharing a security environment, you can now start the remaining CA Top Secret systems as usual.

Chapter 13: Alternate Server Start Options

This section contains the following topics:

Server Alternate SYSRES

CA Top Secret allows for various alternatives to the startup of the service machine. This allows the customer to set standards unique to their own needs.

XAUTOLOG is the mechanism used to start the service machine at IPL time. By the use of the parameter options of the XAUTOLOG command we allow the additional keywords defined by CA Top Secret. To use these commands you must first know two special characters as used by the virtual machine (normally AUTOLOG1) that starts CA Top Secret in your environment. Although the defaults can be changed in your environment in most cases the ESCAPE character will be " and the LINEND character will be #. To determine what your machine uses simply enter CP QUERY TERMINAL and examine the results. For all examples in this chapter we will assume the use of " and # are valid.

CA Top Secret keywords used are:

- VCUU
- PARMNAME
- BACKUP

CA Top Secret supports the use of up to four SYSRES minidisks for the service machine that can be used for whatever purposes you want. These are defined in the directory as the 0100 (required), 0101, 0102, 0103 disks. The 0101 through 0103 disks are optional. Each newly defined minidisk requires formatting, nucleus generation, parameter file creation, and LMP key file creation before it is used. These steps are clearly defined in the Installation Guide as tasks KVC0I035, 061, 065, and 070.

Note: Each disk must get a copy of the LMP key file and parameter file. We recommend that a comment line be added to the front of each TSS PARMS parameter list identifying where the particular file resides.

Examples

For example:

```
* TSS PARMS from 101 disk
```

This comment line will show on the server console at start-up time to clearly identify the file being used.

To start a CA Top Secret server machine using an alternate SYSRES, you must use the following syntax:

```
XAUTOLOG TSSVM"#VCUU 10n
```

TSSVM

Replaced by the name of your CA Top Secret service machine.

"

Terminal escape character.

#

Terminal line end character.

n

Replaced by the minidisk address you wish to IPL.

For example:

```
XAUTOLOG TSSVM"#VCUU 101
```

This command starts up a machine called TSSVM using the 101 disk as the SYSRES.

Note: The CA Top Secret Parameter file that resides on the disk whose address follows VCUU on the command is the parameter file that directs CA Top Secret until a TSS MODIFY(SHUTDOWN) is issued. All restarts will IPL the device specified by the VCUU parameter.

Server Alternate Parameter File

CA Top Secret allows you to use alternate parameter files when starting the service machine. This would be necessary when there are multiple systems sharing the same SYSRES volume for multiple copies of CA Top Secret. The use of an alternate file name allows the site to identify which parameter file should be used. As with alternate SYSRES it is recommended that a comment card be placed in the parameter file to verify which parameter file is being used.

By default the name of the parameter file residing on the IPL disk is TSS PARMS. To start the CA Top Secret server machine using an alternate paramater file you must use the following syntax:

```
XAUTOLOG TSSVM “# PARMNAME filename
```

TSSVM

Replaced by the name of your CA Top Secret service machine.

“

Terminal escape character.

#

Terminal line end character

PARMNAME

Required keyword

filename

Replaced by the file name of the alternate parameter file. The file type of the alternate must be PARMS.

BACKUP

See chapter 11 of this guide.

Chapter 14: PAM Server Support

This section contains the following topics:

PAM Server on z/VM Systems

This chapter describes how to install and configure the PAM Server on a VM system. For information on the PAM Client Server see the PAM Server for Linux for zSeries Getting Started.

CA-ESM Release 1.1 or above, a component of CA-CIS, needs to be installed before attempting to use the PAM server.

You must have openvm extensions running in your SFS system to successfully use PAMSERVE.

Note: The TSS command functions as a CP command and is automatically upper cased by the operating system. To use mixed case values a REXX EXEC must be written or you must use the CASE MIXED option of TSSCRIPT to retain the mixed case characters.

Configuring the PAM Server

Configuring the PAM Server on a VM system requires setting up the PAM service machine and the setting of some options.

Step 1: Create the PAM Server Service Machine

The PAM Server needs to run in a service machine, normally called PAMSERVE.

Create a VM directory entry for the PAM Server service machine. The PAM Server service machine needs to have at least a 1 cylinder 191 minidisk or equivalent SFS space. Below is a sample for the directory entry:

```
USER PAMSERVE pampass 24M 48M G
IPL CMS PARM AUTOOCR
MACHINE ESA
OPTION ACCT MAXCONN 00032 QUICKDSP DIAG88
CONSOLE 0009 3215 T
SPOOL 000C 2540 READER A
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK TCPMAINT 0198 0198 RR
LINK TCPMAINT 0591 0591 RR
LINK TCPMAINT 0592 0592 RR
MDISK 0191 3390 xxxx 1 vvvvvv MR readpw writepw multipw
```

Create a standard TCP/IP PROFILE EXEC similar to your other TCP/IP service machines, and place it on the PAMSERVE 191 minidisk. A sample PAMSERVE PROFILE:

```
/*  PAM Server service machine PROFILE EXEC   */
'ACCESS 198 D'
'ACCESS 591 E'
'ACCESS 592 F'
queue "EXEC TCPRUN"
```

Step 2: Add the PAM Server to SYSTEM DTCPARMS

Define the PAM Server as a TCP/IP service machine by adding an entry in your SYSTEM DTCPARMS on the TCPMAINT 198 minidisk:

```
.* PAM server (PAM) daemon
:nick.PAMSERVE  :type.server    :class.pam
:nick.pam      :type.class
                    :name.PAM daemon
                    :command.SRVRPAM
                    :runtime.C
                    :diskwarn.YES
                    :anonymous.NO
                    :ESM_Enable.Yes
                    :ESM_Validate.RPIVAL
                    :ESM_Racroute.RPIUCMS
                    :VMLINK.CAIMAINT 291 (NONAMES
```

Note: On the CAIMAINT 291 disk you can see a sample RPIVAL routine. It is your responsibility to provide the RPIVAL MODULE for the use of the PAMSERVE machine.

Step 3: Modify the TCPIP Configuration file

Modify your TCPIP Configuration File, normally on the TCPMAINT 198 minidisk, as follows:

1. Add an entry in the AUTOLOG section for the PAM server:

   ```
   PAMSERVE password          ; PAM Server
   ```

2. Add an entry in the PORT section for the port to be used by the PAM server.

   For example to use port 1091:

   ```
   1091  TCP   PAMSERVE                    ; PAM Server
   ```

Step 4: Create the PAM Server Configuration File

The PAM Server needs a configuration file named PAMD CONF on the PAMSERVE 191 if a CMS file is used, or pamd.conf if a BFS file is used. This section describes the options that can be specified. You need to create this file with at least the userid statement/

You can define the options in any order. If you specify an option more than once the last value is taken.

None of the keywords are case sensitive, but the values are. Make sure that you enter things like file names, including the directory portion, in the correct case for the values.

If an option has a default value it is documented. If the default value is the desired value, you do not need to specify that option in the configuration file.

The following options can be specified in the pamd.conf configuration file:

Threads

Specifies the maximum number of threads the PAM Server can start. The default is 32.

Userid

Specifies how to handle the mapping of the Linux for zSeries user ID VM security. Valid values are:

- LINUX—Requires that there is a user map record to convert the Linux for zSeries name to a VM security ID. If the mapping record does not exist the logon fails. The default is LINUX.
- MVS—Bypasses the user map record and tries to validate the user ID passed directly to the VM security product if the user ID is less than 8 bytes. If the user ID is greater than 7 bytes the logon fails
- MIXED—Maps the Linux for zSeries user ID to a VM security ID. If the mapping exists then that user ID is used for validation. If the mapping does not exist and the user ID is 8 bytes or less it will try to perform the validation using the passed in user ID.

Note: Do not change this setting after configuring the types of long user Ids supported. If you make a change after the user ID is added to the /etc/passwd file on the Linux for zSeries machine using PAM, the PAM Client will not find the “old” values to update. It will add the user ID as a new user.

Host

Specifies the address of the interface over which the serve is to accept connections. This value is optional

```
host network-address
```

Where network-address specifies a domain name or an IP address in dotted decimal notation. If a domain name is specified, the server will convert it to an IP address.

If this option is specified the server will only accept connection requests from the interface address specified. If this option is omitted then the server will accept connection requests from all interface addresses configured for this host.

Debug

Specifies the level at which debugging statements and operations statistics should be system logged (currently logged to the PAM Server virtual console). Debug levels are additive and available levels are listed in the following table:

Value	Debug Information
1	Trace function calls
2	Print out packets sent and received
4	Heavy trace debugging
8	Connection management
16	Traces all socket I/O function calls
64	Configuration file processing

TLSRandomFile

Specifies the file from which the server obtains the initial seed for the pseudo-random number generator (PNG). The server updates this file each time the server starts so that the starting value of the PNG changes each time the server is run.

TLSCertificateFile

Specifies the path and name of the server's certificate. This certificate must be in PEM format. The server sends this certificate to a client so that the client can validate the server.

This option is required to use TSL or SSL for communications with clients.

Note: If you do not specify a server certificate and the associated private key, no SSL conversation can be started. If SSL_Required is specified on the Linux client then no communications will ever be started.

TLSCertificateKeyFile

Specifies the path and name of a file that contains the secret private key that matches the certificate stored in the TLSCertificateFile file. This file must be in PEM format. If this file is password protected the server prompts for the password at the time that the server reads the configuration file.

Except as noted below this option is required to use TLS or SSL for communications with clients.

If the server certificate and the associated private key are stored in the same file, this option can be omitted.

Note: Since you run the server as a disconnected service machine, prompting for a password is not possible. In this case you should ensure that the private key is not password protected.

TLSCACertificateFile

Specifies the path and name of a file that contains the certificate for all Certificate Authorities that can sign a client certificate. Each certificate must be in PEM format.

If you have a single CA certificate that is used to sign all client certificates, just specify this file with this keyword. If you have more than one CA certificate, concatenate them together into a composite file and specify the path and name of this composite file with this keyword.

TLSVerifyClient

Specifies whether a client is required to present a certificate when attempting to establish a SSL or TLS connection with the server. Valid values are:

NEVER—The server does not request a certificate. This is the default.

Note: The values OFF, NO, or FALSE are accepted and are equivalent to NEVER.

ALLOW—The server requests a certificate. If no certificate is provided the session proceeds normally. If a bad certificate is provided it is ignored and the session proceeds normally.

TRY—The server requests a certificate. If no certificate is provided, the session proceeds normally. If a bad certificate is provided, the session terminates immediately.

DEMAND—The server requests a certificate. If no certificate is provided, or a bad certificate is provided, the session terminates immediately.

Note: The values HARD, ON, YES, and TRUE are accepted as DEMAND.

Step 5: Create the SRVPAM EXEC

Create the SRVRPAM EXEC on the PAM Server 191 minidisk. This exec contains the OPENVM RUN command that starts the PAM server module LXPAMD.

There are two required parameters and one options parameter used by LXPAMD:

```
LXPAMD –f config_file -p port [-d debug_level]
```

```
-f config_file
```

Specifies the configura5tion file to use for startup. This file can be a variable length record CMS file on the 191 minidisk, or it can be a BFS file. The name can be anything you setup.

The suggested value for BFS:

```
-f pamd.conf
```

The suggested value for CMS file:

```
-f “//PAMD CONF”
```

```
-p port
```

Specifies the startup TCP/IP port that it is running with.

`-d debug_level`

Specifies the level of debug and tracing messages to generate. The value can be from 0 to 65535. The default value is 0.

For example, to start PAM using the CMS file PAMD CONF and port 1091, create a SRVRPAM EXEC with the following lines:

```
/* SRVRPAM EXEC - Start the PAM Server */
'GLOBAL LOADLIB SCREERUN SCEEILBO'
'OPENVM MOUNT /../VMBFS:SFS3XA10:ROOT/ /'
'NUCXLOAD LXPAMD'
'OPENVM RUN LXPAMD -f "//PAMD CONF" -p 1091 '
```

An OPENVM Mount command for the BFS root is required.

If any BFS files are used by the PAM server, an OPENVM SET DIRECTORY command to see the current directory is required.

Step 6: Define the PAM Server ID to security

The following commands are an example used to define the PAM Server user ID and started task information in the CA Top Secret database:

First define a PAM group ACID:

```
TSS CREATE(PAMGROUP) NAME("PAM Server Group") TYPE(GROUP) DEPT(OMVSDEPT)

TSS ADD(PAMGROUP) GID(nn)
```

Now create the server ACID:

```
TSS CREATE(PAMSERVE) NAME("PAMSERVE Service Machine") PASS(password,0) DEPT(deptname)
TYPE(USER) FAC(VM) NODSNCHK NORESCHK

TSS ADD(PAMSERVE) UID(0) GROUP(PAMGROUP) DFLTGRP(PAMGROUP)

TSS PERMIT(PAMSERVE) IBMFAC(ICHCONN) ACCESS(ALL)

TSS PERMIT(PAMSERVE) VMMD(TCPMAINT.,CAIMAINT.0+91) ACCESS(READ)

TSS MODI(OMVSTABS)
```

Configuring CA Top Secret for Use With the PAM Server

The following steps describe how to configure CA Top Secret for use with the PAM Server.

Step 1: Define Linux Nodes to Security

All of the Linux nodes must be defined in the CA Top Secret Security database as NDT node elements. Use the following syntax for each node:

```
TSS ADD|REM|REP(NDT) LINUXNODE(node_name) [IPADDR(ip_address)
FACILITY(facility_name) ACTIVE(YES|NO)]
```

Note: The use of ADD, REM, or REP is required

LINUXNODE(node_name)

Specifies the Linux system name. The maximum length is 246 mixed case characters.

IPADDR(ip_address)

Specifies the IP address for the Linux system. This is treated as a prefix.

FACILITY(facility_name)

Specifies the facility name to use for the system entry validation

ACTIVE(YES|NO)

Indicates whether the node is active and whether to perform system validation.

YES—Node is active and system validation is preformed

NO—Node is inactive and no system validation is performed. This is the default.

To see all defined Linux nodes enter:

```
TSS LIST(NDT) LINUXNODE(ALL)
```

Step 2: Defining the Linux for zSeries User Mappings

Using the new LINUXNAM keyword, define the users Linux for zSeries user name. This allows the mapping of a long name to the users 8-byte security ID.

For those clients that will use the same 8-byte user ID for Linux as they use for their mainframe security ID this step can be skipped but you must configure the PAM Server to support this.

```
TSS ADD|REM|REP(user_acid) LINUXNAM(linux_username)
```

Where linux_username specifies the Linux user name. The maximum length is 1024 mixed case characters. To see a user's current LINUXNAM enter:

```
TSS LIST(user_acid) SEGMENT(LINUX)
```

Step: 3 Define the facilities for the Linux Nodes

You will need to create facilities for the Linux nodes. A single facility name can be used for multiple Linux nodes or you can use individual facility names for Linux nodes. See the Control Options Guide for information on how to name a facility. The facility name associated with the node must be added to the user to authorize him to use that Linux node.

Step: 4 POSIXMGRP control option

Ensure the POSIXMGRP control option has a value other than the system default of zero.

Appendix A: CP Command Attributes Table

This appendix contains the CP command attributes and the appropriate CA Top Secret release.

CP Command	VMUSER Supported	Implied VMPRIV	Operand Included in Security Check
ACNT	No	Yes	None
ACTIVATE	No	Yes	None
AT	No	Yes	Sysname
ATTACH	No	Yes	First
AUTOLOG	Yes	Yes	None
BACKSPAC	No	Yes	First
BACKWARD	No	Yes	First
CACHE	No	Yes	None
CHANGE	Yes	*	Device Name
COMMIT	No	Yes	None
CLOSE	No	No	First
COUPLE	Yes	No	None
CPTRAP	No	Yes	First
DEATIVATE	No	Yes	First
DCP	No	Yes	None
DEDICATE	No	Yes	None
DEFINE	No	No	First
DEFSEG	No	Yes	First
DEFSYS	No	Yes	First
DESTAGE	No	Yes	None
DETACH	No	No	First
DISABLE	No	Yes	First
DISCARD	No	Yes	None
DISPLAY	No	No	First

CP Command	VMUSER Supported	Implied VMPRIV	Operand Included in Security Check
DMCP**	No	Yes	None
DRAIN	No	Yes	First
DUMP	No	No	First
ENABLE	No	Yes	First
FLUSH	No	Yes	First
FORCE	Yes	Yes	First
FORWARD	No	Yes	First
FREE	Yes	Yes	Second
GIVE	No	Yes	First
HALT	No	Yes	First
HOLD	Yes	Yes	Second
INDICATE	No	No	First
IPL	No	No	First
LOADBUF	No	Yes	First
LOADVFCB	No	No	First
LOCATE	No	Yes	First
LOCK	No	Yes	First
MESSAGE	Yes	No	None
MIGRATE	Yes	No	None
MONITOR	No	Yes	First
MSG	Yes	No	None
MSGNOH	Yes	Yes	None
NETWORK	No	Yes	First
NOTREADY	No	No	First
ORDER	Yes	*	Device Name
PURGE	Yes	*	Device Name
QUERY	No	No	First
QVM	Yes	Yes	None
READY	No	No	First

CP Command	VMUSER Supported	Implied VMPRIV	Operand Included in Security Check
RECORDING	No	Yes	None
REDEFINE	No	No	First
REPEAT	No	Yes	First
RETAIN	No	Yes	None
SAVESEG	No	Yes	First
SAVESYS	No	Yes	First
SEND	Yes	No	None
SET	No	No	First
SHUTDOWN	No	Yes	None
SMSG	Yes	No	First
SPACE	No	Yes	First
SPMODE	No	Yes	First
SPOOL	No	No	First
SPTAPE	No	Yes	First
START	No	Yes	First
STCP	No	Yes	None
STORE	No	No	First
TAG	No	No	First
TERMINAL	No	No	First
TRANSFER	Yes	*	Device Name
TRSAVE	No	Yes	None
TRSOURCE	No	Yes	None
UNDEDICATE	No	Yes	None
UNLOCK	No	Yes	None
VARY	No	Yes	First
VMRELOCA	Yes	Yes	Destination (sysname)
WARNING	Yes	Yes	None
WNG	Yes	Yes	None
XAUTOLOG	Yes	No	None

CP Command	VMUSER Supported	Implied VMPRIV	Operand Included in Security Check
XLINK	No	Yes	None
XSPOOL	No	Yes	None

Note: If a command does not appear in the table, then the following default attributes are used:

VMUSER Supported : No, Implied VMPRIV: No, Operand Included in Security Check: None

Appendix B: RACF Compatibility Mode

The RACF Compatibility Mode feature of CA Top Secret is provided as a migration aid for converting from any RACF compatible security product to CA Top Secret. When running in RACF Compatibility Mode, CA Top Secret provides administrative command support for the creation of the CA Top Secret Security File only, while the RACF compatible security product continues to provide resource protection for the VM system. Once the Security File has been created, the migration to CA Top Secret can be completed by regenerating your CP nucleus to replace the RACF compatible security product's CP modules with the equivalent CA Top Secret modules.

Security administration may be performed by entering TSS commands, either from CMS or by submitting TSSCRIPT jobs to the CA Top Secret server virtual machine. However, administrators that will be entering TSS commands from CMS must first establish a communications link with the CA Top Secret server virtual machine, since CA Top Secret has not been generated into the CP nucleus. To do this, the administrator's userid must be authorized to establish an IUCV connection with the CA Top Secret server virtual machine userid, using the CP directory IUCV statement. Once the administrator's userid has IUCV authorization, the communications link is established and the TSS command is defined as a nucleus extension by entering the TSSRCMD command.

Shown below is a sample terminal sesssion highlighting this process:

```
--> TSSRCMD
   Ready;
--> TSS LIST(ALL) DATA(XAUTH)
   TSS9909I IUCV connection accepted by Top Secret
   TSS0130I ADMIN1 Last-used 05 NOV 92 15:14 System=SYSA Facility=VM
   TSS0131I Count=00017 Mode=Warn Name=ADMINISTRATOR
   ACCESSORID = *ALL*    NAME       = GLOBAL-RESOURCES
   XA MINIDISK= MAINT.0190
     ACCESS  = READ
   XA MINIDISK= MAINT.019E
     ACCESS  = READ

   TSS0300I  LIST     FUNCTION SUCCESSFUL

   Ready;
--> TSS PERMIT(ALL) VMMDISK(SYSADMIN.0399) ACCESS(READ)

   TSS0300I  PERMIT    FUNCTION SUCCESSFUL

   Ready;
--> TSS LIST(ALL) DATA(XAUTH)
   ACCESSORID = *ALL*    NAME       = GLOBAL-RESOURCES
   XA MINIDISK=MAINT.0190                        OWNER(MASTER)
     ACCESS  =READ
   XA MINIDISK=MAINT.019E                        OWNER(MASTER)
     ACCESS  =READ
   XA MINIDISK=SYSADMIN.0399                      OWNER(MASTER)
     ACCESS  =READ

   TSS0300I  LIST     FUNCTION SUCCESSFUL

   Ready;
```

Note the following:

- The TSSRCMD command needs to be executed only once during each IPLof the CMS environment.
- When running in RACF Compatibility Mode, the TSS command exists, after execution of the TSSRCMD command, in the CMS environment only. In other words, the CP TSS command does not exist.