CA Top Secret® for z/OS

Implementation: Other Interfaces Guide

r15

Third Edition

CA Technologies Product References

This documentation set references the following CA products:

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

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Documentation Changes

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

- Controlling the Use of a Job Class (see page 30)—Added this topic to discuss authorizations you can make to take advantage of new available SAF calls that control job class usage.

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

- Required STC Definitions (see page 49)—Added information about how system address space started tasks should be defined (to bypass security).
- PSB Security (see page 144)—Corrected syntax in the examples.
- DBD Security (see page 145)—Corrected syntax in the examples.
- Implement Security for CA Datacom (see page 244)—Corrected a DTUTIL resource description and added descriptions for other DTUTIL resources that provide protection for CA Datacom, CA Dataquery, and CA Datacom Datadictionary.

Contents

Chapter 4: Implementing Security for DFSMS 79

Chapter 5: Implementing Security for IMS 93

Chapter 1: Implementing Security for Batch

This section contains the following topics:

About the BATCH Facility

CA Top Secret views BATCH as a facility that must be protected and authorized for use. Each batch job must be associated with an ACID and a password so that CA Top Secret can tell which resources it can access and how they can be accessed. To CA Top Secret, a batch job's ACID is exactly like a user ACID, in that it has an associated user record with a set of specific access authorizations. In fact, it is usually assigned a TYPE(USER) parameter specification when created. All system entry restrictions that can be designated for a user ACID are available for a batch job ACID.

Any user requiring access to BATCH must be explicitly authorized to access this facility. Users are given access to BATCH through a TSS CREATE entry when initially defining a new ACID to CA Top Secret, or through a TSS ADD entry for an ACID already defined to CA Top Secret.

Examples: granting access

This example gives users access to BATCH through when initially defining a new ACID:

```
TSS CREATE(USER01) TYPE(USER)
                   NAME(USER01)
                   DEPARTMENT(TECHDPT)
                   PASSWORD(WHISPER)
                   FACILITY(BATCH)
```

This example gives users access to BATCH for an ACID already defined:

```
TSS ADDTO(USER01) FACILITY(BATCH)
```

BATCH Considerations

The BATCH facility is the easiest facility to address because the resource requirements for these jobs are already defined. In BATCH, the most common resources protected are data sets, volumes, and programs.

Combining the IMPLEMENT mode with one of the ACID derivation capabilities may offer the most flexibility in implementing the BATCH facility. CA Top Secret derives an ACID for BATCH jobs submitted through an online facility, such as TSO, CA-Roscoe, CICS, or IMS. This allows the batch job to run with the same ACID as the ACID of the online user. This is an effective approach to protecting batch testing for online users.

Batch Job Options

CA Top Secret has a number of options to control how a batch job accesses the system. These include:

- Defining ACIDs through the DEFACID sub-option

  You can define a default ACID for the BATCH facility using the DEFACID sub-option of the FACILITY control option. Then define an ACID, or derive an ACID from the reader/terminal from which the job was submitted. This derived ACID is applied to any job without a valid ACID. Combining the derived ACID with IMPLEMENT mode lets you gradually define your batch processing. As you define a valid ACID for each job, the job no longer picks up the default ACID.

  Remove the default after you have completed your BATCH facility implementation to ensure that new batch processing cannot be introduced without being evaluated for the appropriate security controls.

- Deriving ACIDs from the JOB statement

  To initially minimize required JCL revisions, set the JOBACID control option to derive an ACID from information on the existing JOB statement. The use of JOBACID is recommended as an implementation strategy to be replaced with specific ACIDs when the BATCH implementation is complete.

Protecting BATCH Jobs

The major consideration in implementing the BATCH facility is controlling access from batch jobs. Define a specific ACID for each batch production job or each group of batch production jobs. Code a USER= parameter on the JOB statement for each job in your production JCL library.

You then have the flexibility of specifying the appropriate security for each job in your site. You can:

- Group jobs together under the same ACID without tying yourself into other information on the JOB statement which might not be appropriate
- Specify special ACIDs for critical processing, such as payroll master file updating
- Modify the security required for a specific job

You do not have to determine which ACID is derived for a job based on your control option settings. The appropriate ACID is coded on the JOB statement.

You can use derived ACIDs as a permanent strategy in protecting batch processing, or you can migrate to specific ACIDs coded for each job. The derived ACID or the specifically coded ACID must exist or the batch job may abend.

BATCH Passwords

It is inconvenient to change a password in the BATCH facility. If you define a password that expires, you might cause an unexpected interruption in your batch processing. Specifying a password that does not expire protects batch processing if your batch processing is submitted through an online facility.

Production Scheduling

CA Top Secret allows online users to submit batch jobs that run under an ACID other than the ACID of the online user (a secondary ACID). The user's ability to submit jobs that run under secondary ACIDs is the key to effective security for production scheduling. You do not have to allow the production scheduling area to access all production resources at the required access levels. Permit the schedulers to submit batch jobs for the ACIDs you have defined for each production job. If you have specified a password for each ACID, you will have the protection of a password and will not have to divulge the passwords to the schedulers.

This strategy can be applied to automated scheduling software. You can permit the ACID under which the automated scheduler runs to use the ACIDs defined for your production processing. This lets you control what the automated scheduler can submit.

Important! Do not use the attribute NOSUBCHK to give the schedulers (live or automated) the ability to submit production processing. This attribute allows them to submit under any ACID–including critical systems ACIDs and emergency ACIDs. You will also not be able to audit the schedulers' ACIDs to track the production submitted by each scheduler.

Batch Job Definition

When defining a batch job to CA Top Secret, each job submitted for execution must be identified by an ACID and a password.

The ACID is identified through information on each job card, or through a derived ACID.

Password specification is via the PASSWORD= parameter on the job card. CA Top Secret automatically inserts a password on the job card for jobs submitted from other users, based on the ACID of the submitting job, TSO user or STC. The mechanism for automatically inserting this password is the TSS SUBACID control option. For jobs submitted from physical readers, RJE, and NJE terminals, the PASSWORD= parameter must be hard-coded on the job card unless the ACID is created without a designated password (NOPW).

Online Versus Non-online Submission

A batch job can enter the system through either an internal or external reader. When the job is submitted through an internal reader it is described as being submitted *online,* even when the job is being submitted by an unattended batch job or started task. All jobs submitted through an internal reader are processed by CA Top Secret at both job submission and job execution (initiation). If a job is submitted on a node that is not running CA Top Secret and is routed for execution to a node that is, none of the submission time features of CA Top Secret are available for this job. Therefore CA Top Secret does not insert USER= or PASSWORD= on the job card.

External readers are devices like an RJE station or card reader. For an external reader, there is no submission time processing available and no ACID pre-associated with the job unless USER= was hard-coded on the job card.

Assign ACIDs to Batch Jobs Submitted Online

There are two methods for assigning ACIDs to batch jobs submitted online:

- Through the SUBACID control option.

  The SUBACID control option automatically insert a derived ACID into the USER= parameter in the batch job's JOB statement. The value inserted into the USER= parameter is the ACID under which the batch job will run..

- Directly through the USER= parameter on the JOB statement.

  If the USER= parameter on the JOB statement has a value submitted with the job, this value becomes the job's designated ACID and CA Top Secret does not check the SUBACID control option. When you specify an ACID in the USER= parameter of the JOB statement, it overrides the SUBACID control option.

Assign ACIDs to Batch Jobs Not Submitted Online

The information in this section applies to all jobs that are submitted in a BYPASS environment, from a non-CA Top Secret node, or through an external reader. In all of these cases, an ACID must be assigned to the batch job.

There are several different methods for assigning an ACID to a batch job being run from a physical reader, RJE, or non-CA Top Secret secured NJE node. The method you select applies to all batch jobs. The processing order for a given batch job follows the sequence in which the methods are listed below.

Note: Once an ACID is assigned, the rest of the methods are skipped.

- An ACID is assigned by hardcoding it on the job card using the USER= parameter.
- An ACID is assigned by propagating the submitting userid, if the job enters the system via NJE and if the results of the NODES resource check allows propagation.
- An ACID is assigned using the JOBACID control option.
- The global default is used as the ACID. (In certain cases, the job source can be used as the ACID.)

The NODES Resource Class

When a batch job is submitted on a secure z/VM system and routed via RSCS to MVS using an RSCS-to-JES NJE link, CA Top Secret can assign the submitting z/VM user ID to the batch job.

To assign a user ID, enter:

```
TSS PERMIT(ALL) NODES(nodename.USERJ.)
                ACCESS(CONTROL)
```

Note: The NODES resource class replaces the VMJESLNK control option. For example, VMJESLNK(ABCD) is replaced by the syntax:

```
TSS PERMIT(ALL) NODES(ABCD.USERJ.)
                ACCESS(CONTROL)
```

The JOBACID Control Option

In general, the JOBACID control option specifies how to obtain the ACID. This option provides a standard source for deriving the ACID without having to change the existing JCL. JOBACID is used when no user parameter is provided, or when the provided user is not a defined ACID.

If the JOBACID options are not suitable for your site, consider using the Installation Exit.

Specify the JOBACID Control Option

The security administrator can use the JOBACID control option to indicate which field on the job card must be used as the ACID. There are several different JOB statement parameters from which CA Top Secret can derive the ACID for the job:

- A specified field (1-8) of the accounting parameter.

  To illustrate, specifying JOBACID(A,1) indicates that the first parameter of the accounting field must be used as the ACID. For example, ADMIN is the ACID when specified as:

  ```
  //A  JOB (ADMIN,ADM100).
  ```

  The default for this control option extracts the ACID from the first field of the accounting parameter. For example, JOBACID(A,1) indicates the first accounting field is to be used- generating an ACID of 1STACCT. JOBACID(A,1,3) still indicates that the first accounting field is to be used for an ACID name, but the ACID name should be taken starting from the third position in the field- generating an ACID of TACCT.

  For sites that use sub-accounting, the slash (/) and the dash (-) are treated as delimiters of an accounting number used as an ACID. For example, JOBACID(A,1) indicates that in the following account specifications, the ACID is ADM200:

  ```
  ADM200-SMYTHE
  ADM200/JUN82
  ```

- A specified number of characters (1-8) starting from the left of the jobname.

  Here, specifying JOBACID(J,5) indicates that the first five characters of the jobname be used as the ACID. For example, BUG24 is the ACID when specified as:

  ```
  //BUG24MAY JOB 12A,'BUDGET-SMITH'
  ```

- A specified number of characters (1-8) of the source reader's name (other than INTRDR).

  For example, specifying JOBACID(R,8) indicates that the entire reader name be used as the ACID.

If the ACID is not defined to CA Top Secret, it is failed immediately unless a default ACID option (DEFACID) was implemented.

To deactivate JOBACID processing entirely, specify:

```
JOBACID(U,x)
```

Replace x with any numeric value; 7 is recommended.

Override the JOBACID Designation

When the USER= parameter is coded on the JOB statement or if the submitting user is eligible to be propagated, it overrides any ACID derived by the JOBACID specification. If the USER= parameter is coded, its value is used as the ACID assigned to the job, even if the JOBACID specifies another source for the ACID. For example, when USER=ADA103 is coded on the JOB statement and JOBACID(A,2) is specified, the job's ACID becomes ADA103, not the account number.

If the USER= parameter or propagated user is an undefined ACID, CA Top Secret tries to derive an ACID using JOBACID before attempting to use the BATCH facility DEFACID.

Assign the Default ACID

If CA Top Secret is unable to derive an ACID from the USER= parameter or the JOBACID control option, it next checks to see if a default ACID has been assigned. An established default for the ACID allows jobs that otherwise trigger a security violation to run under minimum access authorizations.

Note: If the submitting ACID was propagated from the sending node it overrides the JOBACID.

Establish a Global Default ACID

To use the DEFACID suboption of FACILITY to establish a global default ACID to assign to batch jobs enter the following control option into the Parameter File:

```
FACILITY(BATCH=DEFACID(BATDEF))
```

The BATDEF ACID must be defined with the BATCH facility, have no password specification, and be permitted the appropriate access authorizations. An example showing the creation of a BATDEF ACID is:

```
TSS CREATE(BATDEF) TYPE(USER)
                   NAME(BATDEF)
                   DEPARTMENT(SOFTDEV)
                   FACILITY(BATCH)
                   PASSWORD(NOPW,0)
```

Job Source as the Default ACID

The security administrator can assign a default ACID derived from the name of the physical reader, RJE, or NJE node from which the job is being submitted instead of assigning a single global default. Using the job source (JES device) name as the default ACID allows you to control remote job submission without changing your JCL.

To implement this option, enter the following control option into the Parameter File:

```
FACILITY(BATCH=DEFACID(RDR*TERM))
```

Once the previous entry is made in the Parameter File, the reader's default ACID can be created. For example, to define the default ACID for RJE TERMINAL R10.RD1, the following command is entered with the appropriate access authorizations:

```
TSS CREATE(R10@RD1) TYPE(USER)
                    NAME('DEFAULT-LOC-10')
                    DEPARTMENT(DEPTX)
                    PASSWORD(NOPW,0)
                    FACILITY(BATCH)
                    SOURCE(R10.RD1)
```

RJE devices may now be defined to JES with names up to four digits, for example R1234.RD1. When the RJE node number is four digits then the name for the default ACID is derived by eliminating the period (.) in the device name. Similarly, the SOURCE definition has the period dropped as well. Thus the command to create the ACID in this case would be:

```
TSS CREATE(R1234RD1) TYPE(USER)
                     NAME('DEFAULT-LCO-1234')
                     DEPARTMENT(DEPTX)
                     PASSWORD(NOPW,0)
                     FACILITY(BATCH)
                     SOURCE(R1234RD1)
```

Protecting Card and Remote Readers

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. Because CA Top Secret cannot insert the password on remote or physical readers, the following suggestions are offered for implementing controls in this situation.

- Use a TSS CREATE/ADD entry with the batch job's ACID to specify the PASSWORD(NOPW) attribute. In this example, the JOB1 ACID does not require a password, so password authorization is not checked. JOB1 executes without coding a password in the JOB statement, thus avoiding security violation.

  ```
  TSS CREATE(JOB1) TYPE(USER)
                   NAME(JOB01)
                   DEPARTMENT(OPERAT)
                   FACILITY(BATCH)
                   PASSWORD(NOPW,0)
                   SOURCE(R10)
  ```

  Note: The disadvantage of specifying the PASSWORD(NOPW) attribute is that no CA Top Secret security is enforced for the person who placed the JOB in the reader.

- To allow some control of job submission from remote readers, use the TSS CREATE/ADD entry to attach the TERMINAL attribute to the ACID associated with the batch job. This restricts the job's execution to that device.

  Adding the SOURCE parameter to the ACID also restricts the job's initiation to a specific terminal or reader:

  ```
  TSS ADDTO(JOB1) SOURCE(R10)
  ```

- Another option is to monitor job submission through the AUDIT attribute. A security administrator, with the correct administrative authorities, can add the AUDIT attribute to the ACID associated with the batch job. This provides a trail of all activity related to this ACID in the Audit/Tracking File. For information, see the *Report and Tracking Guide*.

Explicit Password Specification

If explicitly specifying the password on the JOB statement is acceptable from a security standpoint, this can be done using the PASSWORD= keyword as shown in the following example.

```
//REPORT JOB (ACC158),USER=TCH707,PASSWORD=G3T1Q
```

Explicit password specification on JOB JCL statements is a security exposure. Exposures occurs as part of your JCL library, which may be accessed unauthorized, and as spool data.

When JES(VERIFY) is specified:

- If a USER is present in the JOB statement the USER is checked to see if submitter is cross-authorized
- If submission is allowed password processing does not take place unless the password is explicitly coded

Change a Password on a Job Card

To change a password through BATCH, enter:

```
//CHANGE JOB (ACC158),USER=TCH707,PASSWORD=(G3T1Q,FOR2D)
```

Early Password Verification—JES2 SP 3.1.3 and JES3 SP 3.1.3 and above

The early verification is performed unconditionally at the job submission node regardless of the JES control option setting. Unlike earlier releases of JES2 and JES3:

- Early verification is part of the base processing
- Verification occurs at the point of submission

ACID Propagation

If USER= and PASSWORD= are omitted from the job card, JES propagates the submitting ACID to the job. JES associates the submitting ACID with the job and will not require a password on the job card.

Propagation functions as it does under previous JES releases; however, in a 3.1.3 environment JES will propagate the ACID across nodes. Therefore, it is possible to have a job submitted on one node and executed with the submitting ACID on a different node without having USER= or PASSWORD= appear on the job card.

By default, the ACID will propagate; however, by using the PROPCNTL resource class you can suppress ACID propagation, whether or not there are any permits for those resources. This is useful when jobs are submitted from a multi-user online system and your site intends to prevent the region ACID from being associated with batch jobs submitted from the online system.

Examples: ACID propagation

This example suppresses propagation for any ACID starting with CICS:

```
TSS ADDTO(any acid) PROPCNTL(CICS)
```

This example allows propagation to occur after issuing the above command:

```
TSS REMOVE(anyacid) PROPCNTL(CICS)
```

This example suppresses all propagation:

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

This does not affect the normal CA Top Secret SUBACID processing.

Enhanced Submission Time Validation-Under JES2 SP 3.1.3 and JES3 SP 3.1.3 and Above

Additional validation occurs for an ACID's ability to submit a job based upon the combination of ACID jobname and execution node. This validation utilizes the JESJOBS resource class. The JESJOBS resource name is composed of:

- The literal SUBMIT
- The execution node followed by a period
- The jobname followed by a period
- The ACID name

Using this format, if the TSUSER ACID is submitting this sample job (using a JES2 route card):

```
//TESTJOB JOB USER=FRED
/*XEQ FARNODE              (JES2 format)
//IEFBR14 EXEC PGM=IEFBR14
```

The submitting ACID would require the following permit:

```
TSS PERMIT(TSUSER) JESJOBS(SUBMIT.FARNODE.TESTJOB.FRED)
                   ACCESS(READ)
```

Authorization to Submit Batch Jobs

By default, a defined user is only allowed to submit those batch jobs for which the ACID is specifically authorized (jobs identified by his/her ACID). Explicit authority is required to PERMIT a user to submit jobs identified by other ACIDs. Job submission authority is granted via the ACID parameter of the TSS PERMIT command function. For example, to authorize USER01 to submit jobs that run under the direct access authorities of the ACIDs PAYPROD and USER02, enter:

```
TSS PERMIT(USER01) ACID(PAYPROD,USER02)
```

Explicitly permitting a user's job submission authorizations rather than the user's job access authorizations has many useful applications. In the previous entry, PAYPROD is the ACID associated with the batch job that updates the Payroll Master File. The PAYPROD ACID is given all the direct access authorizations required to perform this function. The USER01 ACID, however, is only permitted to run the batch job without being given the authorizations necessary to update the Payroll Master File. Thus, one user is allowed to submit a job for another without sharing access authorizations.

Jobs Submission Through Job Schedulers

Production Job Scheduling Systems (such as CA-Scheduler®) can be given full authority to submit any job as required. This authority can be:

- Unrestricted
- Limited to a specified list of jobs
- Restricted by facility
- Restricted by privileged program

To give the Job Scheduler unrestricted job submission authority, the ACID under which it runs must have the NOSUBCHK attribute attached to it through a TSS CREATE/ADD entry. Since this attribute bypasses all job submission security checking and auditing, it is *not* the recommended procedure.

To provide control over the Job Scheduler's job, the ACID can be limited to a certain facility and/or privileged program.

Examples: job submission

In this example, the ACID JSACID is permitted to access ACID1 only through the CASOMS HA$PSUBS privileged program.

```
TSS PERMIT(JSACID) ACID(ACID1)
                   PRIVPGM(CASOMS HA$PSUBS)
```

Note: When using JES2 3.1.3 and above, HA$PSUBS must be included as a privileged program if there are any PRIVPGM values in use.

Use the PERMIT command function to specify a list of ACIDs that a Job Scheduler is authorized to submit.

In this example, JSACID is the ACID assigned to the Job Scheduling system. ACID1, ACID2, and so on, are the ACIDs associated with the batch jobs that the scheduler is authorized to permit.

```
TSS PERMIT(JSACID) ACID(ACID1,ACID2,...ACIDn)
```

Securing JES Related Resources

In addition to the JESJOBS resource class, an equivalent resource class or functionality is provided for the RACF resource classes shown in the following table.

RACF Resource Class	CA Top Secret Resource Class	Comments
FACILITY	IBMFAC	Used to secure IBM facility resources. For example: TSS PERMIT(USER01) IBMFAC(IDC.DIAG)
JESINPUT	JESINPUT	Used to secure JES job entry sources checked during job initiation. For example: TSS PERMIT(USER01) JESINPUT(SRCNODE)
JESJOBS	JESJOBS	Used to secure submission and cancellation of jobs. For example: TSS PERMIT(USER01) JESJOBS(SUBMIT.MYNODE.MYJOB.MYACID)
JESSPOOL	JESSPOOL	Used to secure JES SPOOL data sets (both SYSIN and SYSOUT) in JES2 and JES3. For example: TSS PERMIT(USER01) JESSPOOL(MYNODE.USER01.JOB1.STC001.D01)
NODES	NODES	Used to control incoming NJE jobs and output, and to assign a userid as the job's or output's owner. For example: TSS PERMIT(ALL) NODES(*.USERS.) ACCESS(READ) NJEACID(&SUSER).
OPERCMDS	OPERCMDS	Used to secure JES and MVS operator commands. For example: TSS PERMIT(USER01) OPERCMDS(MVS.DISPLAY.ACTIVE)
SURROGAT	ACID	To PERMIT an ACID to submit a job for another ACID, use the ACID parameter of the TSS PERMIT command. For example: TSS PERMIT(USER01) ACID(USER02)

RACF Resource Class	CA Top Secret Resource Class	Comments
WRITER	WRITER	Used to secure monitoring, routing, and processing of job output to the local devices, RJE stations, or NJE nodes. For example: TSS PERMIT(USER01) WRITER(JES3.NJE.SITE2)

For information on JES resources, see the JES2 or JES3 *Initialization and Tuning Guides*.

NODES Check

The NODES resource class is used to control incoming NJE jobs and SYSOUT. It can also be used to assign a userid as the owner of the job or SYSOUT on the receiving system.

Controlling the Use of a Job Class

In z/OS 2.1, JES2 and JES3 have new SAF calls to control the use of specific job classes. These SAF calls are triggered when users are permitted to either of the following IBMFAC class resources:

JES.JOBCLASS.OWNER

Checks whether the execution owner has access to the job class.

JES.JOBCLASS.SUBMITTER

Checks whether the submitting user ID has access to the job class.

When users are permitted to these resources, JES2/JES3 issues resource checks for new JESJOBS resources. The new JESJOBS resources have the format (JOBCLASS.*node.class.jobname*).

If you decide to implement these controls, we recommend that you permit users to the JESJOBS resources before activating the controls with permits to the IBMFAC resources.

Example: Restrict Use of a Job Class for a Specific User

Issuing the commands in this example controls job class usage as follows:

- USER1 can submit jobs only for class B on node N67.
- All other users can submit jobs in any class on node N67.

```
TSS ADD(dept) IBMFAC(JES.JOBCLASS.OWNER)
TSS ADD(dept) IBMFAC(JES.JOBLCASS.SUBMITTER)
TSS ADD(dept) JESJOBS(JOBCLASS.N67)
TSS PER(USER1) JESJOBS(JOBCLASS.N67.B)
TSS PER(USER1) IBMFAC(JES.JOBCLASS.OWNER)
TSS PER(USER1) IBMFAC(JES.JOBCLASS.SUBMITTER)
```

dept

Specifies the department ACID to which you are assigning ownership of the resource. This department ACID is associated with the user for which you want to control job class usage.

The Data Lookaside Facility (DLF)

The purpose of the Data Lookaside Facility (DLF) is to control the loading of selected data sets into ESA hyperspaces by selected jobs. CA Top Secret can identify and control those data sets and jobs that are eligible for DLF by using a DLF record that is a special or reserved ACID. This record maintains a list of data sets and jobnames that are eligible for DLF processing.

Note: The FDT defines the format of the data on the DLF ACID.

For example, to indicate that data set CICS.MASTER.FILE should be loaded into a hyperspace upon opening, use the syntax shown below.

```
TSS ADDTO(DLF) DSNAME(CICS.MASTER.FILE)
```

With this entry, the file will be loaded into a hyperspace when the data set is opened. To indicate that the same data set should be loaded into a hyperspace, but only when opened by the CICS region CICSA (as opposed to any batch job), use this syntax:

```
TSS ADDTO(DLF) DSNAME(CICS.MASTER.FILE)
               JOBNAME(CICSA)
```

To indicate that the data set is no longer eligible for DLF processing, use the command shown below.

```
TSS REMOVE(DLF) DSNAME(CICS.MASTER.FILE)
```

To tell MVS to leave the data set in hyperspace when the job ends, use the RETAIN keyword. The syntax appears below.

```
TSS ADDTO(DLF) DSNAME(CICS.MASTER.FILE)
               RETAIN
```

The RETAIN keyword can be added to an existing entry by specifying the same DSN, and it can also be removed. (RETAIN, like JOBNAME, is an optional keyword.) To grant users access to these data sets, use the DLFCLASS keyword. For example, you might ADD the CICS.MASTER.FILE data set to the OPERDEPT Department using the following command:

```
TSS ADDTO(OPERDEPT) DLFCLASS(CICS.MASTER.FILE)
```

You would then PERMIT individual users within that group by using the TSS PERMIT command as shown in the following example.

```
TSS PERMIT(USER01) DLFCLASS(CICS.MASTER.FILE)
```

For more information on the Data Lookaside Facility, see the *User Guide*.

Chapter 2: Implementing Security for STC

This section contains the following topics:

About STC

Once CA Top Secret is installed, default definitions for all STCs have the TSS BYPASS attribute. When explicitly supplying STC protection, use the following considerations as guidelines:

- Required STC definitions must be applied (covered in Required STC Definitions).
- It is recommended that you protect STCs that either reference sensitive data or affect system integrity. In addition, you have the option of protecting whatever is appropriate at your particular site. At a minimum, no other STCs have to be defined.
- It is not necessary to protect STCs that do not reference sensitive data. Also, it is not necessary to protect STCs that run authorized programs which bypass security.

With CA Top Secret, you are able to provide the ideal situation for your community, whatever the scope or degree of STC security is required. STCs can be protected by three levels of security:

- ACID association
- Password
- Operator accountability

Grant Access to the STC Facility

Use of the STC facility is prohibited by CA Top Secret unless the user is authorized access. To do this, the ACID associated with the STC must be created with FACILITY(STC) to allow execution as a started task. Use a TSS CREATE or ADD entry, specifying FACILITY(STC). To ensure that the ACID is only used for the started task and cannot be executed otherwise, do not add any other facilities to that ACID.

Started Task (STC) Considerations

The operations area usually has extensive procedures to make operators responsible for the actions they take.

No matter how extensive the operations procedures are, STC control should be included in the implementation strategy.

You can:

- Define a procedure to the STC Record with a fixed ACID associated with each START. Depending on whether the ACID has a significant password you can either:
  - Require the ACID operand password to assure sensitive procedures cannot be started without authorization.
  - Assign NOPW to the STC ACID to rely on the physical security of operations and operator command security for console entry.
- Define a default policy for procedures undefined to the STC record.
- Define a procedure to the STC Record to prompt for an execution ACID and PASSWORD, when the authority for different instances of the same procedure may differ with each START command.

You can pre-prompt the operator for an STC entry accountability (STCACT) ACID and password to identify the operator who entered the START command. This is in addition to execution security provided by the STC-assigned, default, or prompted ACID.

STC Mode Strategy

The STC facility lends itself to WARN mode implementation because it has its own built-in implementation mode feature for all STCs not defined to CA Top Secret. This lets you define the STCs as you progress, allows the undefined STCs to be controlled by the default, and lets you analyze the violations without impacting the operators. In WARN mode you can choose to suppress the messages sent to the user, which in this case is the operator console.

Once you have refined the definitions of all appropriate started tasks, you can migrate the entire facility to FAIL mode.

Operator Prompts

An important consideration in protecting started tasks is to decide whether operator prompts are acceptable and how extensively you want to use them. STC operator prompts give you additional security on started tasks but they can slow down operations staff. Operators must also be trained in their use. Decide which STCs, if any, are critical enough to warrant operator prompts.

You can provide security on STCs without using operator prompts.

Operator prompts can be used prompt for:

- The ACID under which the started task will run
- The password (if the ACID is defined with a password)
- The operator for his own ACID and password to establish operator accountability to SMF and/or the Audit/Tracking File

Undefined STCs

An important consideration is how to handle started tasks not defined to CA Top Secret. You can choose to bypass security for undefined STCs, bring undefined STCs under the control of a default ACID, or prompt the operator for an ACID under which each undefined STC will run.

Setting up a default for undefined STCs lets you selectively protect started tasks even in FAIL mode. Some started tasks have very little to impact your operation.

If you choose a strategy of bypassing security for undefined STCs, you are exposed by new started tasks that are introduced by your systems or operations staff, because these STCs will run under the bypass default. If this is the strategy selected, establish other controls over the introduction of new started tasks.

Operator Accountability

A powerful STC control available with CA Top Secret is operator accountability. You can record an audit trail of which operators are issuing what STCs. This can be applied to all started tasks or to selected critical started tasks. You can even set this control for undefined STCs.

Define Started Tasks

Started tasks (STCs), by default, are not protected by CA Top Secret. To prevent security exposure, it is prudent to protect important started tasks. Security protection for started tasks is provided once they are defined to CA Top Secret.

To define a started task, the STC must be associated with a specific ACID, action, or both. Once an STC is defined, it is listed in the STC Record which maintains a list of defined STC procedures with their designated ACIDs. Consequently, the Security Records associated with this STC, including its resource authorizations and its password, can be referenced whenever it is executed.

First, create the ACID associated with the STC, using the TSS CREATE command function. Next, use the ADD(STC) function to add the associated ACID to the STC Record through the following operation:

```
TSS ADDTO(STC) PROCNAME(stcname)
               ACID(acidname)
               [STCACT ]
```

stcname

Is either the STC procedure name or the keyword DEFAULT, which is used to establish default processing for all undefined started tasks.

acidname

Is a specified ACID or one of the following keywords used to indicate the designated actions:

- BYPASS-Bypass security
- UNDEF-Treat as undefined user
- FAIL-Fail any undefined STC
- PROMPT-Prompt the operator to supply an ACID and password

STCACT

Requests that the operator supply an ACID and password to provide operator accountability.

When the STC is defined, an STC entry is created in the CA Top Secret STC Record allowing for security validation.

Note: To issue TSS ADDTO(STC) commands, MISC9(STC) administrative authority is required. This means that the security administrator must have MISC9 specified in the ACID's Security Record by a TSS ADMIN command. MISC9(STC) authorizes the maintenance and listing of the STC Record.

Define an O/S Started Task to the STC Record

When JES is started, one or more JCL library concatenations are defined to resolve and expand procedure name references by JCL EXEC statements and O/S console START commands. Jobs which require procedure expansion already have an assigned USER on the JOB statement.

CA Top Secret uses a special STC record to assign security to started tasks initiated from the START command.

STC record entries consist of the keyword information:

PROCNAME

An exact procedure name, corresponding to a JES-defined procedure library member; a procedure name prefix, which matches the initial characters of a procedure library member; or the pseudo-procedure member DEFAULT, which defines the ACID or action assignment for started tasks whose procedure does not match any defined PROCNAME on the STC record.

ACID

Defines a predefined ACID on the security file or the action:

- FAIL—Terminate the matching started task
- BYPASS—Do not assign an acid to the started task
- PROMPT—Prompt for an ACID and PASSWORD assigned to the started task execution

STCACT

(Optional) Indicates that the console operator is prompted to enter an ACID and PASSWORD for the entry of the START command at the console. The ACID entered is only used to log accountability for the START command only. The ACID is not used to assign security to the execution of the started task.

Specifying a Procedure Name

The STC PROCNAME operand specifies either:

- An exact procedure library member name
- A prefix which specifies the initial characters of a matching procedure library member

When the O/S console command START proc1 is entered, CA Top Secret searches the STC Special Record:

- If an exact match is found, the STC PROCNAME assigns the security attributes of the "proc1" STC entry to the expanded JCL which JES supplies from its procedure libraries. If this example had been previously added to the STC, SMURF is assigned to the started task which JES assigned to the START command:

  ```
  TSS ADD(STC) PROCNAME(PROC1)
               ACID(SMURF).
  ```

- If no exact match is found, but one or more prefixes in the STC Record are found, CA Top Secret assigns the STC characteristics of the entry which matches the largest number of initial characters in the START command procedure. In these examples, the best match for START PROC1 is the third entry. ACID STAR4 is assigned to the started task

  ```
  TSS ADD(STC) PROCNAME(P*)
               ACID(STAR)

  TSS ADD(STC) PROCNAME(PRO*)
               ACID(STAR3)

  TSS ADD(STC) PROCNAME(PROC*)
               ACID(STAR4)
  ```

- If no procedure or prefix in the STC record is available, the product searches for a DEFAULT pseudo PROCNAME in the STC record. CA Top Secret makes it possible to assign a number of different action outcomes to DEFAULT assignments. In this example, a procedure which does not match an exact procedure or prefix fails.

  ```
  TSS ADD(STC) PROCNAME(DEFAULT)
               ACID(FAIL)
  ```

- If no DEFAULT PROCNAME has been provided on the STC, the administrator can supply an STC FACILITY DEFACID. For information, see the *Control Options Guide*.

Assign a Specific ACID

An STC is associated with a specific ACID through two TSS command entries- TSS CREATE and ADDTO- that are used to assign a unique ACID to an STC. To associate the STC procedure DUMPSMF with ACID OPS100, you must begin with the CREATE command, as shown below.

```
TSS CREATE(OPS100) TYPE(USER)
                   NAME('DUMPSMF ACID')
                   DEPARTMENT(OPERS)
                   PASSWORD(NUXY,0)
                   FACILITY(STC)
```

In this example, ACID OPS100 is defined with a password of NUXY, which never expires. (Note that the ACID is defined with FACILITY(STC) which allows it to execute as a started task.)

The second CA Top Secret entry required to associate an STC with an ACID is the procedure that adds the STC to the STC Record:

```
TSS ADDTO(STC) PROCNAME(DUMPSMF)
               ACID(OPS100)
```

A family of procedures can be associated with a single ACID by specifying a PROCNAME prefix. For example:

```
TSS ADDTO(STC) PROCNAME(DUMP*)
               ACID(OPS100)
```

This example associates all procnames starting with DUMP with a region ACID of OPS100.

This entry specifies that the DUMPSMF STC is defined and associated with the ACID OPS100, meaning that the DUMPSMF STC procedure executes as a started task under the access authorizations of ACID OPS100. (The authorizations for this ACID are designated through TSS ADDTO(OPS100) and TSS PERMIT(OPS100) entries.) When specified in this way, the operator will never be prompted for the ACID associated with the job, but will have to supply the password NUXY.

STCs with NOPW

Many sites may find it advisable to create STCs with a password of NOPW as shown in the next example.

Note: Specifying a password is required only for critical STCs. In most cases, NOPW is appropriate.

```
TSS CREATE(OPS200) TYPE(USER)
                   NAME('DUMPSMF ACID')
                   DEPARTMENT(OPERS)
                   PASSWORD(NOPW,0)
                   FACILITY(STC)
```

Prompt for ACIDs and Passwords

Because it is possible for several started tasks to execute from the same procedure name, the operator can assign an individual ACID to each instance of the started task procedure. A procedure may be started by different departments with different parameters and security concerns. To resolve this problem, the administrator can assign the action PROMPT instead of a specific ACID for every instance of the procedure. For example:

```
TSS ADD(STC) PROCNAME(IMS*)
             ACID(PROMPT)
```

This example prompts the operator for an ACID and PASSWORD which is assigned to the execution of any procedure which begins with the prefix "IMS":

```
S IMS91
06 TSS7151A Specify AccessorID/Password to be Used With STC=IMS91
R 6,IMS91C/IMSC
IEE600I REPLY TO 06 IS;SUPPRESSED
$HASP100 IMS91 ON STCINRDR
IEF695I START IMS91 WITH JOBNAME IMS91 IS ASSIGNED TO USER IMS91C
 , GROUP OMVSGRP
$HASP373 IMS91 STARTED
```

When the ACID in the STC has NOPW, the operator is not be prompted for the ACID password when starting SMFDUMP with this definition:

```
TSS ADD(STC) PROCNAME(SMFDUMP)
             ACID(OPS200)
```

Record the ACID and Password

A started task may be of such sensitivity that starting the task should record the ACID and PASSWORD of the operator who entered the command. For example, database regions which govern access to sensitive data:

```
TSS ADD(STC) PROCNAME(IMSPAY)
             ACID(PAYRGN)
             STCACT
```

With PAYRGN defined:

```
TSS CREATE(PAYRGN) TYPE(USER)
                   NAME('DUMPSMF ACID')
                   DEPARTMENT(PAYDEPT)
                   PASSWORD(passwd,0)
                   FACILITY(STC,IMSPROD)
```

When an operator enters the console command **START IMSPAY** the message TSS7152A prompts for the operator's ACID and password. When a valid identification is given TSS7150A prompts for the PAYRGN password, which the definition associates with the started task.

Multiple Started Tasks

Because it is possible for several started tasks to execute from the same procedure name, the operator can assign an individual ACID to each instance of the started task procedure. A procedure may be started by different departments with different parameters and security concerns. To resolve this problem, the administrator can assign the action PROMPT instead of a specific ACID for every instance of the procedure. For example:

```
TSS ADD(STC) PROCNAME(IMS*)
             ACID(PROMPT)
```

This example prompts the operator for an ACID and PASSWORD which is assigned to the execution of any procedure which begins with the prefix "IMS":

```
S IMS91
06 TSS7151A Specify AccessorID/Password to be Used With STC=IMS91
R 6,IMS91C/IMSC
IEE600I REPLY TO 06 IS;SUPPRESSED
$HASP100 IMS91 ON STCINRDR
IEF695I START IMS91 WITH JOBNAME IMS91 IS ASSIGNED TO USER IMS91C
 , GROUP OMVSGRP
$HASP373 IMS91 STARTED
```

When the ACID in the STC has NOPW, the operator is not be prompted for the ACID password when starting SMFDUMP with this definition:

```
TSS ADD(STC) PROCNAME(SMFDUMP)
             ACID(OPS200)
```

Record the ACID and Password

A started task may be of such sensitivity that starting the task should record the ACID and PASSWORD of the operator who entered the command. For example, database regions which govern access to sensitive data:

```
TSS ADD(STC) PROCNAME(IMSPAY)
             ACID(PAYRGN)
             STCACT
```

With PAYRGN defined:

```
TSS CREATE(PAYRGN) TYPE(USER)
                   NAME('DUMPSMF ACID')
                   DEPARTMENT(PAYDEPT)
                   PASSWORD(passwd,0)
                   FACILITY(STC,IMSPROD)
```

When an operator enters the console command **START IMSPAY** the message TSS7152A prompts for the operator's ACID and password. When a valid identification is given TSS7150A prompts for the PAYRGN password, which the definition associates with the started task.

Prompt for an Assigned ACID

If different instances of the same procedure can be started with different ACIDs with different authorities, the administrator can have the system prompt the operator for the correct ACID to run the task.

If the ACID assigned to the STC is defined with a password, then the user receives a second prompt for the password.

To allow operators, systems programmers, and production control personnel to start STCs that prompt for the ACID under which they are to run, use this model:

```
TSS ADDTO(STC) PROCNAME(stcname)
               ACID(PROMPT)
               [STCACT]
```

Now, when an operator starts this STC, the operator is prompted for the ACID supplied on the CREATE command and under which the STC is to run. If the ACID has a password, the operator is also prompted to provide it. If the ACID/password combination is valid, STC runs under the authorizations associated with this ACID. If an undefined ACID or an incorrect password is used, then the STC fails immediately.

When STCACT is designated, an initial TSS7152A message prompts the operator for their own ACID and password. This provides accountability for entering the START command for this task. The accountability ACID and password are usually separate from the PROMPT ACID and password under whose authority the task will execute. The accountability ACID cannot have NOPW as a password.

Bypassing Security Checking

To define a specific STC on which no CA Top Secret security checking is performed, a single TSS ADDTO entry is needed:

```
TSS ADDTO(STC) PROCNAME(DUMPSMF)
               ACID(BYPASS)
```

The BYPASS keyword specified in the ACID name field means that started task DUMPSMF is executed without any security checking. To make this type of entry, MISC9(STC) authority is required.

Notes:

- The STCACT attribute cannot be specified for STCs in bypass mode.
- Any STC that accesses USS/OMVS cannot run under BYPASS acid. It must run under a define acid.

STC Accountability

Operators, systems programmers, and production control personnel continuously have access to the O/S consoles and started tasks. Each day, hundreds of STCs may be executed without any record to indicate who entered the started task.

To provide operator accountability, CA Top Secret allows a security administrator to force the operator executing the STC to provide identification. The security administrator with MISC9 administrative authority can ADD an STCACT attribute to the STC definition.

The STCACT attribute forces the operator to enter an accountability ACID and PASSWORD. Normally, this is the ACID and password of the console operator, however administrators may provide for different ACIDs to correspond with their auditability needs.

If the ACID or password entered is invalid, the STC does not EXECUTE, and an audit trail of the failure is logged to the SMF or AUDITx files.

If the ACID and password are valid, an audit trail is logged with both the ACID and password supplied for STC accountability, and with the ACID assigned for started task execution . For information on logging and reporting, see the *Report and Tracking Guide*.

The example assigns the ACID IMS to all started tasks whose procedure name begins with the characters "IMS" and prompts the operator for identity:

```
TSS ADD(STC) PROCNAME(IMS*)
             ACID(IMS)
             STCACT
```

When the optional STCACT attribute is designated the operator is prompted for their ACID and password. This provides accountability for the task's START command. The START command accountability ACID and password is usually separate from the PROMPT ACID and password under whose authority the task executes. The START command accountability ACID cannot have NOPW as a password.

Security Options for Undefined STCs

There are five options for treating undefined STCs. All are specified through a TSS ADDTO(STC) PROCNAME(DEFAULT) entry, and the option you select will apply to all undefined STCs.

Default STC options are:

- Assign a default ACID
- Prompt undefined STCs for user ACID
- Fail all undefined STCs
- Bypass security for all undefined STCs
- Run as an undefined user

Assigning a Default ACID

To establish a default ACID, the security administrator enters:

```
TSS ADDTO(STC) PROCNAME(DEFAULT)
               ACID(acidname)
               [STCACT]
```

All undefined STCs then run under the authorities associated with this ACID. If an executing STC makes a resource access request that exceeds the authority of this ACID, a security error results. The advantage of this approach is that it can be used to allow basic security protection with a minimum impact on productivity.

Prompting for the ACID

To force the operator to supply an ACID for an undefined STC, the security administrator enters:

```
TSS ADDTO(STC) PROCNAME(DEFAULT)
               ACID(PROMPT)
               [STCACT]
```

The STC then executes under the authorities designated for this ACID by the console operator.

Failing All Undefined STCs

To instruct CA Top Secret to fail all undefined STCs, the security administrator enters:

```
TSS ADDTO(STC) PROCNAME(DEFAULT)
               ACID(FAIL)
```

Generally, selection of this option is only advisable in environments in which a systematic and comprehensive analysis of started task security requirements has been made. Before using FAIL, ensure that all STCs are explicitly defined.

The STC facility must be in IMPL or FAIL mode for the undefined STC to fail. If the mode is WARN or DORM, the started task will start and run under an acid of *UNDEF*, which is undefined. In WARN and DORM mode undefined users are not denied access to resources.

Bypassing Security Checking

To instruct CA Top Secret to allow all undefined STCs to bypass security checking, the security administrator enters:

```
TSS ADD(STC) PROCNAME(DEFAULT)
             ACID(BYPASS)
```

There must be careful consideration of the potential security exposures before this option is selected. It is recommended that this option be used only while running tests.

Any STC that accesses USS/OMVS cannot run under BYPASS acid. It must run under a define acid.

Running as an Undefined User

To instruct CA Top Secret to allow undefined STCs to run as an undefined user, the security administrator enters:

```
TSS ADDTO(STC) PROCNAME(DEFAULT)
               ACID(UNDEF)
```

Whatever approach has been selected for handling undefined users, also governs how undefined STCs are handled. This approach could also be used to run undefined STCs under a default ACID.

Required STC Definitions

The following are required started task (STC) definitions.

STC	Definition
INIT	If the default STC ACID is BYPASS, then no additional definitions are required. Any valid STC ACID can be used with INIT, so that the starting batch initiator will not log as initiating BYPASS: TSS ADDTO(STC) PROCNAME(INIT) ACID(BYPASS) If the default is FAIL, then an explicit definition must be given for INIT, either as BYPASS or by using an ACID that has a facility (STC). TSS ADDTO(STC) PROCNAME(INIT) ACID(STCACID)
JES2 or JES3	The recommended approach is to explicitly define an ACID for JES (using whatever name is desired) and adding it to the STC record. BYPASS should not be used with JES unless there are no installation-provided JES exits that invoke CA Top Secret and the installation is not using JES(VERIFY). In all cases, it is best to create an ACID for JES. For example: TSS CREATE(JES) NAME('JES ACID') PASSWORD(NOPW,0) DEPARTMENT(STCDEPT) FACILITY(STC) TSS ADDTO(STC) PROCNAME(JES2) ACID(JES)
LLA, VLF, CATALOG and other System Address Spaces	System address spaces should bypass security, so these started tasks should be defined through the following entry: TSS ADDTO(STC) PROCNAME(procname) ACID(BYPASS) If the default STC ACID is BYPASS, then no additional definitions are required. Any valid STC ACID can be used with LLA so that the restarting LLA will not log as initiating BYPASS. See the previous INIT explanation for examples. **Note:** When restarting LLA from an undefined terminal or ACID, two messages are received: TSS7220E: 101 J=lla A=*missing and: CSV244I CSV READ ACCESS DENIED To correct this error, you must either PERMIT SYS1.PARMLIB to the ALL Record, or give access to the data set name for the ID starting the LLA.
TSS, TSSB, TSSBKUP, TSSRVCR1, TSSRESTR, TSSRESTN, SMSRESTR and SMSRESTN	BYPASS should be assigned either by the STC default (if the STC default is BYPASS) or by an explicit definition. Care must be taken to properly define these STCs.

Assigning a MASTFAC Facility to a Multi-User Address Space

When a started task is associated with a multi-user address space, the ACID assigned to the PROCNAME is associated with a MASTFAC attribute.

When a user connects to the address space through VTAM, or other telecommunication method, the MASTFAC assigns a facility from the CA Top Secret FACILITY matrix, which each session's user is required to have. Parameters from the FACILITY matrix provide additional security policies to all sessions connected to the address space.

This example assigns the CICSTEST facility to procedure CICST1 and requires all users signing onto that region to have the CICSTEST facility. CICSPEON connects to procedure CICST1 through its applid, and signs on using CESN with the USERID and PASSWORD 'CESN USERID=CICSPEON,PS=POWRLSS':

```
TSS CREATE(CICST1) TYPE(USER)
                   NAME ('CICS TEST REGION')
                   FACILITY(STC,CICSTEST)
                   PASSWORD(NOPW,0)
                   DEPARTMENT(CICSDP)
                   MASTFAC(CICSTEST)

TSS ADD(STC) PROCNAME(CICST1)
             ACID(CICST1)

TSS CREATE(CICSPEON) TYPE(USER)
                     NAME('LOWLEVEL USER')
                     FACILITY(CICSTEST)
                     PASSWORD(POWRLSS)
                     DEPARTMENT(CICSDP)
```

In this example, CICSPEON can not sign on as its facility CICSTEST does not correspond to the CICSPROD facility associated with ACID CICSP1's MASTFAC:

```
TSS CREATE(CICSP1) TYPE(USER)
                   NAME ('CICS TEST REGION')
                   FACILITY(STC,CICSPROD)
                   PASSWORD(NOPW,0)
                   DEPARTMENT(CICSDP)
                   MASTFAC(CICSPROD)

TSS ADD(STC) PROCNAME(CICSP1)
             ACID(CICSP1)
```

For information, see the *Control Options Guide* and *Implementation Guides*.

Automatic Logon

Automatic logon is supported for STCs (and subsystems) that are started prior to the first initialization of CA Top Secret after an IPL (these STCs later invoke security checking after it has initialized). This feature builds a security environment when the first security event takes place in that address space after TSS has fully initialized.

- CA Top Secret uses the STC Record for automatic logon, as if the started task was initialized after CA Top Secret initialization.
- Automatic logon begins when SVC security events occur. For example: RACROUTE REQUEST=AUTH, and RACROUTE REQUEST=VERIFY, ENVIR=CREATE but not RACROUTE REQUEST=FASTAUTH.
- Automatic logon is intended for primary job entry subsystems (JES2 or JES3), or console automation products such as CA-Opera and NETVIEW.
- Automatic logon should *not* be used for CICS, IMS, CA-Roscoe®, Batch initiators, or other online STCs.

Chapter 3: Implementing Security for APPC z/OS

This section contains the following topics:

APPC z/OS Considerations

The APPC z/OS component is a cooperative processing interface available for MVS/ESA 4.2.0 and above. Cooperative processing refers to the ability of application programs to establish communications with partner programs in other systems and within the same system (for example, when testing is required). This allows the sharing of work, data, and services between systems and across networks. With APPC z/OS, transaction programs (TPs) on one z/OS system can initiate (allocate) conversations with other TPs on different systems throughout an SNA network.

Implementing APPC Security

To secure APPC conversations:

- Review your resources and determine what degree of security you need
- Secure the ASCH, ASCHINT, and APPC started tasks and permit them to the APPC facility for minimal security
- Specify which LUs can establish sessions by:
  - Adding the appropriate parameters to the VTAM APPL statement
  - Defining remote and partner LUs to the CA Top Secret APPCLU record
  - Identifying which sessions are reserved for APPC use
- Determine how TP profile and side-information data sets are maintained (with or without database tokens)
- Secure individual user access to APPC resources by adding new authorization restrictions to the profile records

Cooperative Processing

The APPC/z/OS/OS/390 component is a cooperative processing interface available for OS/390/ESA 4.2.0 and above. Cooperative processing refers to the ability of application programs to establish communications with partner programs on other systems and within the same system (such as when testing is required). This allows for the sharing of work, data, and services between systems and across networks. With APPC/OS/390, transaction programs (TPs) on one z/OS or OS/390 system can initiate (allocate) conversations with other TPs on different systems throughout an SNA network.

Depending on the nature of the TPs, the data being shared, and on the comparative security environments of the partner z/OS systems (that is, one z/OS or OS/390 system is secured by CA Top Secret in FAIL mode while another z/OS system is not secured at all); there is the potential for security exposures.

Note: In an unprotected network, the only information a TP requires to initiate a conversation with another TP is the name of that TP and the Logical Unit (LU) on which that TP is located.

Options

Several options are provided for securing the APPC interface. Which options you choose to use, however, depends on the sensitivity of your resources and the particular security requirements of your systems. To best implement security for an APPC environment, it is extremely important for you know:

- Your exact security needs
- The activities occurring on your secured and unsecured systems
- The different CA Top Secret APPC security options and levels

This chapter addresses the following aspects of the CA Top Secret APPC/OS/390 interface:

- Preventing improper access to secured data and resources during an APPC conversation
- Adding user SYSOUT account information and delivery information to user ACIDs and profile records.

Note: Before continuing, you should already be familiar with the IBM *Planning: APPC Management Guide*.

What is APPC?

An APPC/OS/390 implementation from a single host processor consists of:

- Two started tasks (APPC and ASCH)
- VTAM 6.2 logical units (LUs)
- Transaction programs (TPs)
- One or more TP profile data sets
- A single side-information data set

The purpose of each of these components is described following:

APPC Started Task

Used to manage conversations between TPs.

ASCH Started Task

Used to manage sets of transaction initiators (the address spaces that run the TPs on inbound requests); also called the transaction scheduler.

LUs

Nodes or other points of entry into the network. Each Transaction Program (TP) is associated with a single LU.

TPs

Referenced on the EXEC statements found in the TP profile JCL stream; can be written in COBOL, C, PL/I, FORTRAN, ASSEMBLER, or REXX.

TP profile data set

VSAM KSDS data sets used to describe the TPs available from a single LU or set of LUs reached by an inbound transaction request. It contains the TP class and the JCL used to run the TP in the transaction initiator. TP profile data sets are administered using either a batch utility (ATBSDFMU) or an interactive ISPF dialog.

Side-information file

VSAM KSDS used by the APPC address space to translate symbolic destination node names to the appropriate VTAM LU/TP names. There is one side-information file per system.

What are WORKATTR Fields?

If the TP profile for a particular TP program specifies the TAILOR_ACCOUNT or TAILOR_SYSOUT attributes, APPC is directed to customize the execution environment of the TP program by using values associated with the ACID. This information consists of an account number and a set of fields containing distribution information for SYSOUT produced by the TP. This information, collectively called WORKATTR data, is added to an ACID's user and/or profile record(s).

There are nine WORKATTR fields. See Adding SYSOUT and ACCOUNT Information for information on how to administer these fields- including information on the required administrative authority and examples of their use.

What Security Measures Should I Take?

Security for an APPC/OS/390 environment is discretionary and should be based on the sensitivity of your resources and on the relationship between the conversing systems. If, after careful consideration, you decide that you need to add some security measures, there are several different levels you can employ. They are:

- ASCH, ASCHINT, and APPC STCs (minimum)

  In a minimally secured APPC environment, the ASCH started task must be permitted READ access to SYS1.PARMLIB. The APPC started task requires READ access to SYS1.PARMLIB and UPDATE access to the TP profile and side-information data sets. The ASCHINTS, APPC, and ASCH started tasks must also be granted access to the APPC facility.

- LU-LU (optional)

  Security at the LU-LU level can be provided through a combination of CA Top Secret and VTAM options. By using the APPCLU Record Table you can define which LUs can be used for an APPC conversation and what, if any, security information is required for that link to take place. By using the VTAM VERIFY option and the CA Top Secret VTAMAPPL and APPLICATION resources you can determine which ACBs can be opened to establish a session between authorized LUs and what degree of security checking will take place. CONVSEC security can further extend these security restrictions to the conversation level.

- Individual User Access (optional)

  Security on a TP-to-TP level limits which users have access to certain APPC resources and is provided through CA Top Secret keywords. For example, you can restrict USER01 so that he can only execute those TPs identified by the PERS.database token-you can even force him to execute those TPs from a particular LU.

  Furthermore, the administration of the TP profile and side-information data sets can be limited to a particular individual or individuals. Administration of the APPCLU record can also be limited by using the TSS ADMIN command function.

Implementing APPC Security

There are five basic steps to secure APPC conversations:

- Review your resources and determine what degree of security you need.
- Secure the ASCH, ASCHINT, and APPC started tasks and permit them to the APPC facility for minimal security.
- Specify which LUs can establish sessions by:
  - Adding the appropriate parameters to the VTAM APPL statement,
  - Defining remote and partner LUs to the CA Top Secret APPCLU record, and
  - Identifying which sessions will be used with APPC. The use of the APPCPORT and APPLICATION parameters will help in defining these sessions.
- Determine how TP profile and side-information data sets will be maintained (for example, with or without database tokens).
- Secure individual user access to APPC resources by adding new authorization restrictions to the ACID user and profile records.

The information derived from Step 1 will determine what resources you will need to secure, and which of the remaining steps you will need to take.

Securing APPC STCs

The APPC Started Task manages conversations between TPs. Sessions are established between LU 6.2 nodes dedicated for that purpose. The parameters identifying those LUs are acquired from the APPCPMxx member of SYS1.PARMLIB.

The ASCH Started Task is called the transaction scheduler because it manages sets of transaction initiators- the address spaces that run the TPs named on inbound requests. (These transaction initiators are sometimes referred to as ASCHINTs.)

The APPC address space acquires and validates inbound transactions requests, passing those that are accepted to ASCH. Scheduling parameters residing in the ASCHPMxx member of SYS1.PARMLIB define the TP classes and the number of initiators associated with each.

If you have decided to implement security for APPC, the first thing you need to do is to add the APPC and ASCH started tasks to the CA Top Secret STC record. To do this you need to:

- Define an ACID for each STC.
- Add the ACID and the procedure to the STC Record Table.
- Grant the STC ACIDs access to the APPC facility and the appropriate data set access authorizations.

Defining ASCH, ASHINT, and APPC STCs to the STC Record

Defining the APPC and ASCH STCs to the STC Record Table is a two-part process. You must:

- Define each STC to CA Top Secret by associating it with a specific ACID. In the example below the APPC, ASCH, and ASCHINT user ACIDs are created and associated with the APPC and ASCH started tasks, respectively. By specifying the NOPW operand for the PASSWORD keyword, the requirement for operator prompts to start the STC is eliminated.

```
TSS CREATE(APPC) TYPE(USER)
                 NAME('APPC ACID')
                 DEPARTMENT(OPERS)
                 PASSWORD(NOPW,0)
                 FACILITY(STC,APPC)
```

```
TSS CREATE(ASCH) TYPE(USER)
                 NAME('ASCH ACID')
                 DEPARTMENT(OPERS)
                 PASSWORD(NOPW,0)
                 FACILITY(STC)
```

```
TSS CREATE(ASCHINT) TYPE(USER)
                    NAME('ASCHINT ACID')
                    DEPARTMENT(OPERS)
                    PASSWORD(NOPW,0)
                    FACILITY(STC)
```

- Add the STC to the STC record. Using the above examples, the commands would look like this:

```
TSS ADDTO(STC) PROCNAME(APPC)
               ACID(APPC)
```

```
TSS ADDTO(STC) PROCNAME(ASCH)
               ACID(ASCH)
```

```
TSS ADDTO(STC) PROCNAME(ASCHINT)
               ACID(ASCHINT)
```

These started tasks will now execute under the access authorizations given to the APPC, ASCH, and ASCHINT ACIDs.

Granting Access Authorizations

The next step is to grant the proper access authorizations to these STCs by issuing the appropriate TSS PERMITs for the ACIDs. In addition to the proper data set authorizations, each of these STCs must also be permitted to the APPC facility.

The APPC facility is already defined in the Facility Matrix and operates under the following defaults:

```
INITPGM=ATB       id=AP  TYPE=03
ATTRIBUTES=IN-USE,ACTIVE,NOSHRPRF,NOASUBM,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,NORNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFCMD
ATTRIBUTES=MSGLC,NOTRACE,EODINIT,DORMPW,NONPWR
MODE=WARN  DOWN=GLOBAL    LOGGING=INIT,MSG,SEC9
UIDACID=8  LOCKTIME=000  DEFACID=*NONE*   KEY=8
MAXUSER=03000   PRFT=003
```

Note: To run APPC, users must also be authorized to the APPC facility.

The APPC STC requires READ access to SYS1.PARMLIB and UPDATE access to the TP profile and side-information data sets. Using the APPC ACID created in the previous example, and assuming the SYS1.PARMLIB data set is already owned, you would issue the following command:

```
TSS PERMIT(APPC) DSNAME(SYS1.PARMLIB)
                 ACCESS(READ)

TSS PERMIT(APPC) DSNAME(tpdsname)
                 ACCESS(UPDATE)

TSS PERMIT(APPC) DSNAME(sidsname)
                 ACCESS(UPDATE)
```

You might also consider providing both STCs with the NODSNCHK and NOVOLCHK bypass attributes.

The ASCH STC requires READ access to SYS1.PARMLIB. Using the ASCH ACID created in the previous examples the command would look like this:

```
TSS PERMIT(ASCH) DSNAME(SYS1.PARMLIB)
                 ACCESS(READ)
```

The ASCHINT STC does not require any particular permits.

Allocating Security Statements

When an APPC conversation allocation is issued, that request also indicates what type of security information, if any, will be provided by the inbound TP. The required security information can take one of the following forms:

- SECURITY_NONE-No security information is provided. The LU can either accept the request and run the designated TP in a special execution environment or deny it completely.

  Since SECURITY_NONE requests do not provide a userid, APPC must construct a default execution environment for the TP.

- SECURITY_PGM-The request contains a userid, password and, optionally, a security profile name. System entry validation must be performed on the inbound side.
- SECURITY_SAME-The request contains a userid, security profile name, and the already verified (AV) indicator. APPC will accept the user's identity without performing validation.

The most frequently used form is SECURITY_SAME, especially when requests are issued from one trusted z/OS system to another equally trusted z/OS system. If no userid is provided, however, the request is treated the same as if SECURITY_NONE was specified.

Securing LU-LU Sessions Part 1 - VTAM

For an APPC conversation to take place, you need to define the LUs to VTAM. This is done by coding VTAM APPL statements. At this point you can also use VTAM to verify a TP's authorization to use a particular session and remove inbound conversations that do not provide the appropriate level of security information. This is done by:

- Identifying APPC ACB sessions through the CA Top Secret VTAMAPPL keyword
- Adding security information on the VERIFY and SECACPT keywords of the APPL statement to make VTAM verify LU-to-LU session requests and accept default levels of conversation security between LUs

The VTAMAPPL, SECACPT, and VERIFY keywords are discussed briefly in the next sections. For information, see the IBM *Planning: APPC* Management *Guide*.

Securing Sessions

When a TP on one LU initiates a conversation with a TP on another LU that conversation takes place across a session. Corresponding access method control blocks (ACBs) are opened from the APPC address space when APPC is started on your z/OS system. To prevent non-APF authorized programs from opening ACBs dedicated to APPC processing, and thereby potentially intercepting message traffic destined for APPC, you should define the acbnames to CA Top Secret via the VTAMAPPL keyword. To do so, use the following syntax:

```
TSS ADDTO(owner-acid)  VTAMAPPL(acbname)
```

Note: It is recommended that the owning ACID be that of the APPC department to which the STC ACIDs belong.

If you are aware of certain ACIDs who should not be authorized to use a particular ACB or set of ACBs you can also use ACTION(DENY). For example to restrict USER01 from opening the ACB123 session use the following syntax:

```
TSS PERMIT(USER01) VTAMAPPL(ACB123)
                   ACTION(DENY)
```

VTAM Verify

Regardless of whether or not you have secured ACBs through VTAMAPPL, you can still indicate whether or not VTAM will verify that the partner LU is authorized to establish a session with the host LU. This is specified through the VERIFY parameter on the VTAM APPL statement. There are three operands to choose from:

NONE

VTAM does not have to verify the partner LU; the default.

OPTIONAL

VTAM should verify those partner LUs that have defined LU-LU passwords (known as sesskeys or session keys).

REQUIRED

VTAM must verify every partner LU.

When verification is required, VTAM compares the sesskeys assigned to each LU. These sesskeys can be established when you define authorized LU-LU links to the APPCLU record. For more information on the CA Top Secret SESSKEY keyword, see LINKID Keywords; for information on IBM session keys, see the IBM *VTAM Resource Definition* Reference.

Defining SECACPT Values

The CA Top Secret VTAMAPPL resource class and the VERIFY parameter of the VTAM APPL statement are both used to secure session usage. The SECACPT values on the VTAM APPL statement carry that security one step further by extending it to the conversation level. When you are defining an LU to VTAM, you must specify the greatest level of security to be allowed on inbound requests for TPs at the LU. This is done by specifying one of the following operands on the APPL statement's SECACPT keyword:

NONE

Requests that contain no security information; the default.

CONV

Requests with specified security information.

ALREADYV

Requests with specified security information and requests indicating that security information has already been verified.

Use ALREADYV only between equally secured LUs

PERSISTV

Requests with persistent verification and access security information.

AVPV

Requests with either already verified security information or persistent verification and access security information.

When specifying a value for the SECACPT keyword you should keep in mind the nature of the TPs using that LU. For example, if TPA issues a conversation request with SECURITY_PGM, a userid and password will be provided. Therefore, CONV would be the appropriate SECACPT value for the LU that TPA will be issuing a conversation request to.

The SECACPT value indicates the default level of acceptable conversation security. This value can later be overridden by the CONVSEC value supplied when LU-LU authorized links are defined to the CA Top Secret APPCLU record.

Securing LU-LU Sessions Part 2 - APPCLU

After reviewing your security requirements and determining which LUs will be used for APPC conversations and what level of security should be maintained for those conversations, you need to identify authorized LU links to CA Top Secret. The LINKID keyword is used to identify the authorized LU links.

The APPCLU record maintains a list of LINKIDs (one LINKID per LU-LU connection) in the following format:

```
netid.localLU.partnerLU
```

netid

The name of the network on which the local LU resides.

Note: You should supply the same netid specified by the netid= statement in the VTAM ATCSTRxx member.

localLU

The VTAM name of the local LU.

partnerLU

The VTAM name of the partner LU.

You do not have to specify all three values. However, the algorithm is designed to search for the best match. Therefore the most explicit entry (i.e., the longest) which best matches the requested resource name will be used. The following entries begin with the most explicit example and end with the least explicit example.

```
NET1.SYSTEMA.SYSTEMB
NET1.SYS
NET
```

LINKID and APPCLU

To add an entry to the APPCLU record, use the following syntax:

```
TSS ADDTO(APPCLU) LINKID(netid.localLU.partnerLU)
    [SESSKEY(sesskey) INTERVAL(nnnnn) [CONVSEC(operand)]]

    - or -
    [SESSLOCK]
```

Note: If you have the network qualified names feature active in VTAM, use the four-level name or a prefix. For example:

```
local-netid.luid1.remote-netid.luid2.
```

If you do not have the network qualified names feature active in VTAM, use the three-level name. For example:

```
netid.localLU.remoteLU.
```

These entries can later be REMOVEd through the TSS REMOVE command and updated using the TSS REPLACE command.

LINKID Keywords

The LINKID keyword is used to maintain the appropriate entry to the APPCLU record for APPC conversations. Each LINKID entry can be associated with one or more of the following keywords:

SESSKEY

A 16-byte hexadecimal encryption key unique to each LU-LU authorized link. If additional verification is required for a session to be established between the two LUs, the SESSKEY is used to encrypt and decrypt connection messages. If the exchange is satisfactory to both LUs, the session is established.

Note: If you supply a SESSKEY, you must also indicate an INTERVAL.

SESSLOCK

Used to single out which LUs cannot establish sessions.

INTERVAL

Must be supplied if you indicate a SESSKEY. This keyword determines how frequently SESSKEY must be updated. The value can range from 0 to 32767 where 0 indicates that the SESSKEY does not expire.

CONVSEC

Determines what, if any, additional identification needs to be provided and verified for an LU-LU session to be established. Select one of these values:

- NONE-No security information is required.
- CONV-Security information is required. A userid and password need to be verified.

  Note: CONV overrides the SECACPT value specified in the VTAM APPL statement.
- ALREADYV-Indicates that a userid and password have already been verified by the partner LU and that the path is trusted.
- PERSISTV-Indicates that a userid and password must be verified on the first request; for subsequent requests only the userid is verified.
- AVPV-Supports both ALREADYV and PERSISTV values; the security type used depends on the incoming request.

Note: CONVSEC security does not apply if you are using a VTAM release prior to 3.4.

Examples: the APPCLU record

In the following examples LSCA01 is responsible for maintaining the APPCLU record.

In this example, LU01 and LU02 can establish sessions but the SESSKEY, which is A1B2C3D4, must be verified first. That SESSKEY must also be updated every 45 days. To accomplish this, LSCA01 enters:

```
TSS ADDTO(APPCLU) LINKID(SYS1.LU01.LU02)
                  SESSKEY(A1B2C3D4)
                  CONVSEC(CONV)
                  INTERVAL(45)
```

In this example, LU01 *cannot* establish a session with LU03. To prevent this, LSCA01 enters:

```
TSS ADDTO(APPCLU) LINKID(SYS1.LU01.LU03)
                  SESSLOCK
```

In this example, the SESSKEY used to verify that LU02 and LU03 can establish sessions has to be updated every 30 days. Previously, the SESSKEY only had to be changed every 60 days. To institute this change, LSCA01 enters:

```
TSS REPLACE(APPCLU) LINKID(SYS1.LU02.LU03) INTERVAL(30)
```

In this example, LSCA01 can issue a TSS LIST command to review and verify the changes he has made:

```
TSS LIST(APPCLU) DATA(SESSKEY)
```

The following information is provided:

```
 ACCESORID   = *APPCLU*   NAME         = APPC/MVS LU SECURITY
 LINKID      = SYS1.LU01.LU02
   TOT VIOS  =     0      MAX VIOS     =     0  LINK STATUS=AVAILABLE
   CONVSEC   = CONV       VIOLATIONS   =     0
   SESS KEY  = ABCDEFGH
     EXPIRES = 92-30-11   INTERVAL     =    45
 LINKID      = SYS1.LU01.LU03
   TOT VIOS  =     0      MAX VIOS     =     0  LINK STATUS=UNAVAILABLE
   CONVSEC   = N/A        VIOLATIONS   =     0
   SESS KEY  = N/A
     EXPIRES = N/A
```

This example views a single entry in the APPCLU record:

```
TSS LIST(APPCLU) LINKID(SYS1.LU02.LU03)
```

When only a single entry is listed, the SESSKEY is not displayed. To view the SESSKEY associated with a particular LINKID, you *must* view the entire APPCLU record using the syntax demonstrated in the previous example.

Authority

To maintain the APPCLU record, the security administrator needs to have MISC2(APPCLU) authority. To view SESSKEYS, he must also have DATA(SESSKEY) authority.

SAF Operator Commands for Persistent Verification

Persistent Verification (PV) is a type of conversational signon security that significantly reduces the number of I/Os and updates sent to the Security File and the number of ACIDs and passwords that are transmitted between local and remote LUs. Persistent Verification is used whenever the SECACPT parameter on the VTAM APPL statement is set to PERSISTV. It is also used whenever the CONVSEC parameter on the APPCLU statement for the link is set to PERSISTV.

With PV, a copy of the user's signed-on information is kept in extended CSA storage. Keeping this information available greatly reduces the calls to the Security File. The existing LINKID, SESSKEY, and CONVSEC keywords define the information needed to create session keys and how the session keys are used in APPC/OS/390 conversation.

Two SAF operator commands, CASF DISPLAY and CASF SIGNOFF, are used to monitor the sign-on information that is stored in the ECSA. CASF DISPLAY is used to determine what authorities are being used by various LUs. CASF SIGNOFF is used to remove an ACID's signed-on information from the Sign-On List.

Displaying Signed_On_From List for an LU

To enable administrators to monitor system use, an SAF operator command is supported to display information by local LU (APPL), by remote LU (POE), by userid (USER), by group (GROUP) or by security label (SECLABEL). The syntax of this command is:

```
CASF DISPLAY (APPL|POE|USER|GROUP|SECLABEL)
```

The issuer must have READ authority for the resource name for the OPERCMDS class.

The following example shows the command you would issue to determine who is signed on from the remote LU named L09IX004.

```
CASF DISPLAY POE(L09IX004)
```

In response to this command, the Signed_On_From list for remote LU L09IX004 is displayed. If none of the optional parameters are entered, a list of all the entries in the POE/APPL table will be displayed.

Removing Users From Signed_On_From List

A similar operator command exists to remove users from the Signed_On_From list. The issuer of this command must have UPDATE authority for the resource name for the OPERCMDS class. Although the parameters for the remove command are the same as the parameters for the display command (APPL,POE,USER,GROUP,SECLABEL), three parameters are required: APPL, POE, and USER.

The syntax of the remove command is:

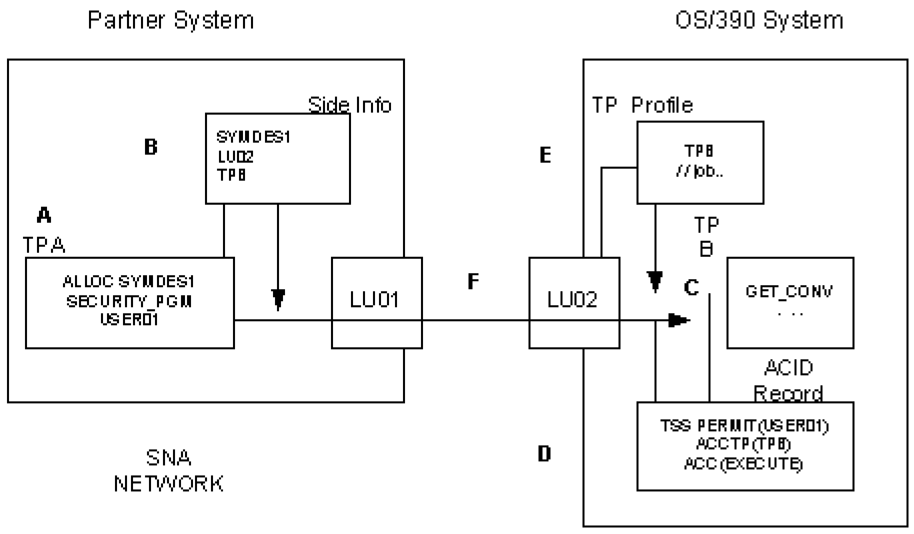
```
CASF SIGNOFF APPL POE USER (GROUP|SECLABEL)
```

The following example shows the command you would issue to signoff user HOWPA02 from local LU L08IX003 and remote LU L09IX004.

```
CASF SIGNOFF APPL(L08IX003),POE(L09IX004),USER(HOWPA02)
```

Administering TP Profile and Side-Information Files

In APPC/OS/390, side information and TP profile files contain routing and scheduling information that z/OS uses to find and initiate TPs in response to allocate requests from other TPs (do not confuse these profiles with the Profile ACID Type). The following figure illustrates how these components are used.



In the previous figure, the following occurs:

A

TPA allocates a conversation with TPB. SECURITY_PGM is indicated and the USER01 ACID is supplied. The symbolic destination name is SYMDES1.

B

APPC compares the SYMDES1 symbolic destination name with the information supplied in the side information file and learns that it refers to TPB and LU02.

C

TPB receives the allocation request.

D

Since SECURITY_PGM is specified on the request, CA Top Secret views the security record for USER01 and learns that USER01 has authorization to EXECUTE TPB.

E

TPB is located in the TP profile data set along with the corresponding JCL and scheduling information.

F

The entire conversation takes place across a VTAM session.

The administrator responsible for maintaining security for APPC resources creates and maintains TP profiles and side information files by using the APPC/OS/390 administration utility (ATBSDFMU) or the APPC/OS/390 administration dialog (an interactive front-end to ATBSDFMU).

The TP profiles and side-information files are stored in VSAM key sequenced data sets (KSDS). Since several TP profiles can be maintained in the same TP profile data set, by PERMITting an ACID the authority to administer one TP profile data set you are PERMITting that ACID the authority to administer all of the TP profiles (and consequently all of the TPs) contained in that data set.

If you want to authorize different ACIDs to administer each TP profile or subset of TP profiles individually, you need to use database tokens.

Securing Database Tokens

A database token is a one to eight-character name that is used to identify an individual file within a data set. Each token can be used more than once. For example, within TPDSN01 TP profile data set you might have the following TP profiles:

```
    Payroll.TPA        Research.TPD
    Payroll.TPB        Research.TPE
    Payroll.TPC        Pers.TPF
```

As you can see, TPA, TPB, and TPC have been grouped together with the Payroll token. TPD and TPE are given the Research token, while TPF is associated with the Pers token.

Although each administrator can be given access to the TP profile and side-information file data sets, if database tokens are used, a second level of security checking is performed by z/OS. For example, John Smith, the administrator for the Payroll Department may be restricted to all TP profiles identified by the Payroll database token. Sue Jones, the administrator for the Personnel Department, may be PERMITed to EXECUTE TPs identified by the Pers token but may only READ TPs identified by the Payroll token.

It is recommended that a single administrator (either an SCA or LSCA) be responsible for creating TP profile and side-information data sets and assigning database tokens, if necessary, to individual files within those data sets. In addition to having access to the data sets, this “super-administrator” should also be restricted to accessing them only through the APPC/OS/390 administration utility. For example:

```
TSS PERMIT(acid) DSNAME(dsnnames)
                 PRIVPGM(ATBSDFMU)
```

To create and maintain database tokens, this administrator would use the APPC/OS/390 administration utility's DBMODIFY command. (For more information about the DBMODIFY utility, see the IBM *Planning: APPC Management Guide*.)

To control access to these tokens once they are created, you can use the existing IBMFAC resource. For example, if APPCADMN is to be responsible for maintaining database tokens, you would issue the following commands:

```
TSS ADDTO(owning-acid) IBMFAC(APPCMVS.)

TSS PERMIT(APPCADMN) IBMFAC(APPCMVS.DBTOKEN)
                     ACCESS(UPDATE)
```

Securing TP Profiles

Security for individual TPs and TP profiles is administered through the APPCTP resource class. The APPCTP resource class name takes the following format:

```
dbtoken.level.tpname
```

dbtoken

The one to eight-character database token associated with the TP profile.

level

Represents one of the following:

- SYS1-If the TP is available to all users who can access the LU.
- groupid-If the TP is only available to a particular group.
- acid-If the TP is only available to a particular user.
- tpname-The one to 64 character TP name.

Authority

The administrator requires access to the APPC facility, READ access to view the TP profiles and UPDATE access to create, modify and delete TP profiles. APPC users also require access to the APPC facility and will need EXECUTE access to the TP profile to run the associated TP.

For example, if LSCA01 is to be responsible for maintaining TPB (which has a database token of PAYROLL and will be accessible to all users for LU02) the following command would be issued:

```
TSS PERMIT(LSCA01) APPCTP(PAYROLL.SYS1.TPB)
                   ACCESS(UPDATE)
                   FACILITY(APPC)
```

For users with the TECHPROF profile to run TPB, the following command would have to be issued:

```
TSS PERMIT(TECHPROF) APPCTP(PAYROLL.SYS1.TPB)
                     ACCESS(READ)
                     FACILITY(APPC)
```

The security administrator for APPC resources (LSCA01 in the examples) should create an APPCTP profile for each TP profile, using generic prefixing where appropriate. ACIDs requiring use of the TPs should then be given the appropriate access.

Securing Side-Information Files

The authorization to access side information files is controlled by using the APPCSI resource class keyword. This keyword is composed of three levels in the following format:

```
dbtoken.SYS1.symbolic-destination-name
```

dbtoken

The one to eight-character database token associated with the side-information file.

SYS1

Indicates that this file is available to all users.

symbolic...

The one to eight-character symbolic destination name associated with the side-information file.

To view the side information, the administrator requires READ access. To create, modify or delete these files, the administrator requires UPDATE access. In both cases, the administrator must have access to the APPC facility. Prefixing applies, but masking does not. Therefore, to allow LSCA01 to maintain all side information files assigned to the RESEARCH database token, issue the following command:

```
TSS PERMIT(LSCA01) APPCSI(RESEARCH.SYS1)
                   ACCESS(UPDATE)
```

Note: ACIDs who use symbolic destination names on an outbound allocate request do not require access to APPCSI profiles.

Specifying ACID Authorization

In addition to the different layers of security that have already been discussed, CA Top Secret also provides a layer of OS/390/APPC security that is specifically designed to limit individual user access to APPC communication paths and TPs. This allows the administrator to:

- Ensure that the user has access to the target LU (via APPLICATION)
- Ensure that the user may originate requests from the source LU (via APPCPORT)
- Ensure that the user is authorized to EXECUTE the TP he requests (via APPCTP)

APPC runs the TP under the security environment built from the security information associated with the ACID provided on the inbound request. This means that the TP will only have access to those data sets and other resources that the user would normally have access to in a batch or TSO address space initiated on the target system.

Controlling Access To and From APPC LUs

When a TP on one LU allocates a conversation request with a TP on another LU, two questions are asked by the target LU. They are:

- Is the TP permitted to access this remote LU?
- Is the TP permitted to access this remote LU from that local LU?

Access to the target LU can be restricted by identifying the application name of the target LU. This is done through the APPLICATION resource. For example, if you want to restrict SMITH01 so that he can only target LU02 from his local port of LU01, you would issue:

```
TSS PERMIT(SMITH01) APPLICATION(LU02)
```

If the APPLICATION resource class is used to designate which users can issue a request *to* a particular LU, the APPCPORT is used to restrict which LU that request can be issued *from.* For example, if you wanted to restrict JONES01 so that he could only allocate a conversation request from LU01, you would issue:

```
TSS PERMIT(JONES01) APPCPORT(LU01)
```

The remote LUs that LU01 can establish a session with are still restricted according to the definitions listed in the APPCLU record.

Securing TPs

Authorization to execute a particular TP is provided by the APPCTP resource class. To secure a TP you must first secure the TP profile that defines it. Once the TP profile is secured, users must be granted READ access to run the TP. For example, the following command allows USER01 to run the TPA transaction program:

```
TSS PERMIT(USER01) APPCTP(PAYROLL.SYS1.TPA)
                   ACCESS(READ)
```

Adding SYSOUT and ACCOUNT Information

You can specify SYSOUT account and delivery information in an ACID's user and profile security records. This information will be used when a Transaction Program (TP) is run on behalf of that user, and can be added to either the user or profile record by using the TSS ADD command function with the appropriate keyword. The TSS REMOVE and TSS REPLACE commands can be used to alter that information once it has been added. Syntax examples are included at the end of this section.

For APPC/OS/390 to know that this information should be extracted from the ACID's user or profile record, the APPC/OS/390 administrator must first use the APPC/OS/390 administration dialog or utility to specify YES on the TAILOR_ACCOUNT and TAILOR_SYSOUT keywords for the appropriate TP. For more information on TAILOR_ACCOUNT and TAILOR_SYSOUT, see the IBM *Planning: APPC Management Guide*.

WORKATTR Keywords

SYSOUT delivery and account number information can be added to an ACID's user or profile record using the fields listed and described below. This information, collectively called WORKATTR data, can be located on an ACID's security record and/or on the security records of any profiles that ACID is associated with. If a WORKATTR request is made against an ACID, his user record will be searched first and then each of his profiles in order, until the appropriate information is found.

WANAME

Specifies the name of the user to whom SYSOUT information is to be delivered.

Maximum: 60 characters

WABLDG

Specifies the building that SYSOUT information is to be delivered to.

Maximum: 60 characters

WADEPT

Specifies the department that SYSOUT information is to be delivered to.

Maximum: 60 characters

WAROOM

Specifies the room that SYSOUT information is to be delivered to.

Maximum: 60 characters

WAADDR1 - 4

Specifies up to four more address lines for SYSOUT delivery.

Maximum: 60 characters line

WAACCNT

Specifies an account number for APPC/OS/390 processing.

Maximum: 255 characters

Note: If TAILOR_ACCOUNT is specified in the TP profile entry, the number indicated by WAACCNT is assigned to the execution of the TP- overriding the account specified in the TP profile JCL.

If spaces are included in the operand you need to enclose that value within single quotes. For example, if Dave Jones is the value for WANAME, specify WANAME('Dave Jones'). See the *Command Functions Guide* for a description of these commands.

Data Authority Operand

In addition to these fields, there is also an operand for the DATA authority called WORKATTR. This operand is specified when performing a TSS LIST command and provides information about the values specified in any of the fields listed previously.

Authority

To add, remove, or update WORKATTR data for the ACIDs within his scope of authority the administrator must have MISC2(WORKATTR) authority.

Examples

In this example, the DCA for the R&D Department would like to have all SYSOUT information generated by ACIDs using the TESTPROF profile delivered to Ed Brown in room 212 of the Second Floor Lab. The DCA would issue the following command:

```
TSS ADDTO(TESTPROF) WANAME('Ed Brown')
                    WAROOM(212)
                    WADEPT('R&D').
                    WAADDR1('Second Floor Lab')
```

In this example, the event that Sharon Brophy takes over Ed's position, the following command can be issued:

```
TSS REPLACE(TESTPROF) WANAME('Sharon Brophy')
                      WAROOM(213)
```

In this example, if Mel Thomas has WORKATTR information included on both his User ACID and on the TESTPROF ACID he is associated with, the information provided by his user record takes precedence.

In this example, if the DCA wanted to list the SYSOUT distribution information for the TESTPROF profile, he would issue:

```
TSS LIST(TESTPROF) DATA(WORKATTR)
```

Chapter 4: Implementing Security for DFSMS

This section contains the following topics:

About DFSMS

The *IBM System-Managed Storage Migration Planning Guide* advises RACF be used to control DFSMS functions. The RACF recommendation is there to advise the DFSMS client that only an external security package can be used to restrict DFSMS activity, and an SAF-compatible external security product (such as CA Top Secret, CA-ACF2, or RACF) is required to protect DFSMS functions.

CA Top Secret's philosophy of being SAF-compatible allows this IBM use of SAF classes to be smoothly integrated into it.

For information on the resources and calls discussed in this chapter, see the *IBM System-Managed Storage Migration Planning Guide*.

DFP 3.0 and DFSMS

DFSMS is IBM's designation for the DF/HSM, DFDSS, DFSORT and RACF products when used in a DFP Version 3 context.

The Data Facility family and RACF provide complementary functions that make up the Data Facility Storage Management Subsystem (DFSMS). The DFSMS components (DFHSM, DFDSS, DFSORT, and RACF) are all separate complementary products using OS/390/DFP services. However, OS/390/DFP does not have any other DFP or non-DFP product as a prerequisite to support its functions. Although RACF has also been designated as part of DFSMS, RACF is not a DFP product. The RACF participation is there so that the SAF calls being issued by the DFP DFSMS products will be processed. Without an SAF-compatible security system (like CA Top Secret, CA-ACF2, or RACF), there is no security for any DFSMS function.

DFP Release 3 is the provider of the Storage Management Subsystem. In the pre-release 3 OS/390 DFP, basic data management definition and services are provided for allocation and enforcement of data and storage attributes. In DFP 3, several enhancements have been added to simplify catalog processing, integrate VSAM into standard storage management functions, and to allow definition and enforcement of storage allocation requirements via a CLIST-like language used for coding data management definitions through the ISMF facility. The concept of absolute device allocation, such as UNIT=3380, is superceded by the concept of generic storage groups and attributes.

Storage Management Subsystem

The Storage Management Subsystem (SMS) is a feature of IBM's DFP Version 3 and above that handles new data set allocations to volumes that it controls, and associates new attributes with those data sets. The Interactive Storage Management Facility (ISMF) provides a mechanism to specify which volumes are to be under SMS control and which data sets will be placed on SMS controlled volumes (that is, which data sets will be SMS controlled). The process SMS uses to determine whether or not the data set being allocated is to be SMS controlled involves classifying the data set in storage management terms.

Important! The storage management classification has no correspondence or effect upon security data classification or security resource classes.

Storage Management Classes

SMS recognizes the storage management data set classes:

Data

Defines attributes about the data set that would normally be associated with the DCB or the AMP parameters in the JCL. Data class definitions take the place of model DSCBs, but also include information for VSAM data sets as well. Even if a data set is not SMS controlled, a Data class can be associated with it.

Management

Defines space and availability management attributes of the data set. This includes how often to back it up and how many backup versions should be kept. This class can only be associated with an SMS controlled data set.

The definitions of all of the classes are kept in a control data set and each of them is known by a unique eight-character name. It is through this name that access to data set classes is secured through external security calls. SMS controlled data sets carry the name of the Data, Storage, and Management classes associated with them in their ICF catalog and VVDS records.

Storage Groups

The DASD volumes that are to be SMS controlled are divided up into Storage Groups that are similar in concept to DASD esoteric unit names. They are similar in that they define a group of volumes, but they are different in the following way:

- Storage Groups have volume maintenance characteristics associated with them.
- All volumes in a Storage Group must have the same device characteristics, such as number of tracks per cylinder and number of bytes per track.
- A given volume can only be in one Storage Group.
- The definition of Storage Groups that are shared among multiple SMS systems are identical because each system shares the control data sets that define them.

Data Set Allocation

When a new data set is being allocated, the classes are selected by the Automatic Class Selection (ACS) routines which are coded by the user in a language similar to TSO CLIST. There are four types of ACS routines: three types which select each of three types of classes and one to select the candidate storage group(s) that will ultimately hold the data set. These routines have access to and can use information about the job, the user, the data set, and the environment to select the appropriate classes. However, since this suppresses the power of active selection, it is not usually recommended.

When SMS is active, all new allocations pass through the Data Class ACS routine to have the Data class assigned and then into the Storage Class ACS routine to see if a Storage class will be selected. If no Storage class is selected, the data set will *not* be SMS controlled and the allocation will continue through normal allocation processing. Prior to allocation, SMS will call the external security manager to check the authority of the selected storage class name.

If a Storage class is selected, the data set will be controlled by SMS, so the allocation will then be passed through the Management Class ACS routine to have its Management class selected. The SMS controlled allocation is then passed to the final ACS routine to have its candidate Storage Group(s) selected.

Note: When defining a data set using JCL, the user may code new JCL parameters to specify the names of the classes, but not the name of the Storage Group.

By associating this new information with the data sets, data set management software can serve the user's needs more specifically. Having system software manage data sets is the basic goal of SMS, but SMS just provides the framework in which to accomplish data set management, not the tools to do it.

DFP Components

The components of DFP Version 3 provide:

- A z/OS subsystem honoring standard subsystem interfaces for communication, and the System Authorization Facility (SAF) for all security controls. (This makes SMS information fully accessible by other system software components.)
- A real-time handler of data set classification (from a storage, not a security, perspective) and new data set allocations on SMS controlled volumes.
- A DASD device grouping facility.
- A facility that stores additional information in the ICF catalog, the VVDS, and in the VTOC entries of SMS controlled data sets.
- A facility that can help reduce JCL coding requirements.
- A facility to define performance preferences for allocations.
- A source of information for system programs that request it.

SMS brings new information about DASD data sets that it manages and provides new services to programs and systems that are prepared to use that information.

Understanding the DFSMS Classes and Attributes

SMS resource protection is provided through four existing classes:

FACILITY

Controls the use of catalog, IDCAMS, and DFDSS functions against SMS managed volumes. This class is administered as class IBMFAC in CA Top Secret to avoid confusion with the CA Top Secret FACILITY concept. These new calls supercede the FACILITY resource name of IGG.CATLOCK previously introduced for controlling catalog access.

PROGRAM

Controls the use of DFSMS programs. This is used in exactly the same manner as program protection with CA Top Secret. The programs invoked for the various DFSMS ISMF functions may be protected as programs by their program name.

STORCLAS

Controls user access to a specific storage class defined by the storage administrator. Each unique storage class is identified by a unique one- to eight-byte name.

MGMTCLAS

Controls user access to a management class defined by the storage administrator. Each unique management class is identified by a unique one- to eight-byte name.

In addition, four ACID attributes for default data class name, default management class name, default storage class name, and default application identifier, have been added to CA Top Secret. These attributes are tested by the DFSMS Automatic Class Selection (ACS) routines to obtain this information. To administer these SMS resources and attributes, the administrator needs MISC2 authority. For information, see the *Command Functions Guide*.

Controlling DFSMS Functions and Commands (FACILITY Resource Class)

The FACILITY class as used by IBM is identified by the name IBMFAC when using CA Top Secret.

A list of DFSMS Function/Command resource values and definitions follows. The values on this list are predefined by IBM for each respective DFSMS Function/Command. A definition of each documented function and two examples showing protection and allowing access are provided. Although extended TSS PERMIT functions such as Prefixing, TIME, and ACTION are not illustrated, they can be used if desired.

Note: An access level of READ is required.

For a more complete list of DFSMS Function/Command resource values and definitions, see the IBM *System-Managed Storage Administration* Reference *Guide*.

DFSMS Function/Command	Definition
STGADMIN.IGD.ACTIVATE.CONFIGURATION	Provides the ability to activate a configuration.
STGADMIN.IDC.DIAGNOSE.CATALOG	Allows IDCAMS DIAGNOSE to run against catalogs.
STGADMIN.IDC.DIAGNOSE.VVDS	Allows access method services diagnose to be run against the VVDS when a comparison against the BCS is being performed. This profile protects the BCS under this circumstance. It does not protect the ability to run AMS DIAGNOSE commands against a VVDS.
STGADMIN.IGG.ALTBCS	Provides the ability to alter BCS (VSAM Basic Catalog Structure) entries.
STGADMIN.IGG.DIRCAT	Provides the ability to use a directed catalog.
STGADMIN.DEFNVSAM.NOBCS	Allows non-VSAM entries to be defined without BCS entries.
STGADMIN.IGG.DEFNVSAM.NONVR	Allows non-VSAM entries to be defined without NVR (non-VSAM Volume Record) entries.
STGADMIN.IGG.DEFNVSAM.NOBCSCHK	Provides the ability to delete an NVR entry without checking the BCS entry.
STGADMIN.IGG.DELETE.NOSCRATCH	Allows a BCS entry to be deleted without deleting the associated VTOC entry and NVR.
STGADMIN.IGG.DELGDG.FORCE	Provides the ability to delete a GDG using the FORCE operand.
STGADMIN.ADR.COPY.BYPASSACS	Allows the DFDSS COPY function to bypass the ACS routines and use the original (pre-DFSMS) constructs.

DFSMS Function/Command	Definition
STGADMIN.ADR.RESTORE.BYPASSACS	Allows the DFDSS RESTORE function to bypass the ACS routines and use the original (pre-DFSMS) constructs.
STGADMIN.ADR.COPY.INCAT	Provides the ability to use the INCAT parameter on a DFDSS COPY function.
STGADMIN.ADR.DUMP.INCAT	Provides the ability to use the INCAT parameter on a DFDSS DUMP function.
STGADMIN.ADR.CONVERTV.INCAT	Provides the ability to use the INCAT parameter on a DFDSS CONVERTV function.
STGADMIN.ADR.CONVERTV	Allows the DFDSS CONVERTV function to be used.

SMS Functions and Commands Administration

The FACILITY class as used by IBM is identified by the name IBMFAC when using CA Top Secret to avoid any confusion between FACILITY as a class and FACILITY as the Facility limitation and control capability of CA Top Secret. The ability to define and limit both signon and resource access by Facility is unique to CA Top Secret and not supported by RACF; therefore, RACF chose to use a different representation for the FACILITY class name.

Examples: FACILITY Authorization

This example protects the function STGADMIN.IDC.DIAGNOSE.CATALOG:

```
TSS ADDTO(deptacid) IBMFAC(STGADMIN)
```

Only a maximum of eight characters is allowed to specify the particular IBMFAC resource with the TSS ADD command function, enter STGADMIN to protect *all* SMS functions beginning with the prefix STGADMIN.

This example allows the function:

```
TSS PERMIT(acid) IBMFAC(STGADMIN.IDC.DIAGNOSE.CATALOG)
                 ACCESS(READ)
```

This example protects the function ADR.DUMP.INCAT:

```
TSS ADDTO(deptacid) IBMFAC(STGADMIN)
```

This example allows the function:

```
TSS PERMIT(acid) IBMFAC(STGADMIN.ADR.DUMP.INCAT)
                 ACCESS(READ)
```

DFSMS Program Control

The following program values are associated with the new DFP 3.0 ISMF functions. From a CA Top Secret perspective, these are real program names and are protected through the PROGRAM resource class in the same manner as any other program.

DFSMS ISMS program names and functions:

ISMF Applications

- DGTFFLAD—Invokes ACS Application
- DGTFSACD—Invokes CDS Application
- DGTFSGDR—Invokes SG Application
- DGTSCB01—Protects Set Cache

ISMF STORCLAS Applications

- DGTFSCAA   STORCLAS—Alter Dialog
- DGTFSCDI STORCLAS—Display Dialog
- DGTFSCLD STORCLAS—List Dialog
- DGTFDIS1 STORCLAS—List Display Line Operator
- DGTFALS1 STORCLAS—List Alter Line Operator
- DGTFCAS1 STORCLAS—List Copy Line Operator

ISMF DATACLAS Applications

- DGTFDCDA DATACLAS—Define Dialog
- DGTFDCAA   DATACLAS—Alter Dialog
- DGTFDCDI DATACLAS—Display Dialog
- DGTFDCLD DATACLAS—List Dialog
- DGTFDID1 DATACLAS—List Display Line Operator
- DGTFALD1   DATACLAS—List Alter Line Operator
- DGTFCAD1 DATACLAS—List Copy Line Operator

ISMF MGMTCLAS Applications

- DGTFMCDA MGMTCLAS—Define Dialog
- DGTFMCAA MGMTCLAS—Alter Dialog
- DGTFMCDI MGMTCLAS—Display Dialog
- DGTFMCLD MGMTCLAS—List Dialog
- DGTFDIM1 MGMTCLAS—List Display Line Operator
- DGTFALM1 MGMTCLAS—List Alter Line Operator
- DGTFCAM1 MGMTCLAS—List Copy Line Operator

ISMF Commands

- DGTFACAT—Activate Command

Examples: PROGRAM Authorization

This example protects the ISMF program DGTFFLAD:

```
TSS ADDTO(deptacid) PROGRAM(DGTFFLAD)
```

This example allows the program:

```
TSS PERMIT(acid) PROGRAM(DGTFFLAD)
```

This example protects the ISMF program DGTFACAT:

```
TSS ADDTO(deptacid) PROGRAM(DGTFACAT)
```

This example allows the program:

```
TSS PERMIT(acid) PROGRAM(DGTFACAT)
```

DFSMS Storage and Management Class Access Control

The STORCLAS and MGMTCLAS resource classes allow you to protect and allow user access to specific storage classes and management classes, respectively.

These storage and management classes are each identified by a unique one- to eight-byte name. It is this name that is used in describing a class to CA Top Secret and thereby protecting it.

The following example shows the TSS commands needed to protect a fictitious storage class of PRODSTOR and a fictitious management class of PRODMGMT.

Note: An access level of READ is implied for each permission although it has no distinguishing effect on access to the resource class.

To protect the storage class PRODSTOR, enter:

```
TSS ADDTO(deptacid) STORCLAS(PRODSTOR)
```

To allow an ACID to use the PRODSTOR storage class, enter:

```
TSS PERMIT(acid) STORCLAS(PRODSTOR)
```

To protect the management class PRODMGMT, enter:

```
TSS ADDTO(deptacid) MGMTCLAS(PRODMGMT)
```

To allow an ACID to use the PRODMGMT management class, enter:

```
TSS PERMIT(acid) MGMTCLAS(PRODMGMT)
```

Automatic Class Selection ACID Attributes

If the value for ACSDEFAULTS is YES, the data set owner, application identifier, and default class information will be extracted from CA Top Secret. If you do not use the input variables to ACS routines that are saved in CA Top Secret, you should set this option to NO to save additional calls to CA Top Secret during allocation processing.

CA Top Secret has four eight-byte ACID attributes (SMSDATA, SMSMGMT, SMSSTOR, and SMSAPPL) to support default data class name, default management class name, default storage class name, and default application identifier attribute, respectively. These attributes can be tested by the DFSMS Automatic Class Selection (ACS) routines to establish the SMS class attributes for a data set.

These attributes may be specified on any User or Control type ACID.

To administer these attributes, an administrator needs MISC2(SMS) authority.

Examples: adding attributes

This example adds a default management class of SYSTCLAS and a default storage class of SYSSTOR to user SYSFRED:

```
TSS ADDTO(SYSFRED) SMSMGMT(SYSTCLAS)
                   SMSSTOR(SYSSTOR)
```

This example adds a default data class of ALLDATA and a default application identifier of PAYROLL to user SYSMIKE:

```
TSS ADDTO(SYSMIKE) SMSDATA(ALLDATA)
                   SMSAPPL(PAYROLL)
```

For information on the ACS ACID attributes, see the *Command Functions Guide*.

Note: While establishment of default data, storage, and management classes, and default application identifier attribute, is completely supportable through CA Top Secret in the same manner as RACF, IBM does not recommend their use for most applications. As stated by IBM in the *System-Managed Storage Migration Planning Guide*, “We do not recommend the use of default classes in the user or group profile because it is highly unlikely that a given SMS class is applicable to all data sets a user creates. We suggest instead that you use ACS to determine the data, storage and management classes, as well as the application identifier attribute.”

RESOWNER Use

During data set allocation processing, DFSMS requests the external security package to derive the RESOWNER of the specified data set name. If the data set is defined to CA Top Secret and the ownership record for that data set has been assigned a specific ACID as the RESOWNER, then this ACID is returned to DFSMS. If there is no RESOWNER assigned to a defined data set, or if the data set is not owned, CA Top Secret determines if the High Level Qualifier matches a User or Control ACID. If the High Level Qualifier is a valid ACID, this ACID is returned as the RESOWNER. In all other instances, the currently signed on ACID will be returned as the RESOWNER.

The RESOWNER attribute can be added to a data set, at the same time ownership of that data set is established. For example:

```
TSS ADDTO(deptacid) DSNAME(PAY.)
                    RESOWNER(smsacid)
```

If a data set is already owned and you want to assign a RESOWNER to that data set, you must determine the owner of the data set and the exact prefix of the data set that is owned. This can be accomplished with the TSS WHOOWNS command function. If you are attempting to assign a RESOWNER to data set PAY.MASTER, then you must initially perform:

```
TSS WHOOWNS DSNAME(PAY.MASTER)
```

A response indicates that PAYDEPT owns DSNAME(PAY.). To assign or change a RESOWNER of smsacid to data set PAY.MASTER, enter:

```
TSS ADDTO(paydept) DSNAME(PAY.)
                   RESOWNER(smsacid)
```

To remove a RESOWNER of smsacid from data set PAY.MASTER, enter:

```
TSS ADDTO(paydept) DSNAME(PAY.)
                   RESOWNER()
```

Note: When removing a RESOWNER there is nothing specified within the parentheses.

Implementation Considerations

Consider how you would like to establish RESOWNERs for data sets. Generally, RESOWNER values should only be added to data set prefixes whose High Level Qualifier does not match a Control or User ACID. If you do not specifically assign a RESOWNER to data set ownership elements, the High Level Qualifier of the data set will be returned as the RESOWNER, if it matches a defined User or Control ACID.

For ACID-related data sets, using the High Level Qualifier as the RESOWNER is probably most desirable, because it is the user who creates and allocates the data set. During allocation processing, if the derived RESOWNER for a data set matches the currently signed-on user, there is less overhead in returning the ACS defaults for this RESOWNER, as well as in the authorization check for the SMS storage class (STORCLAS) and SMS Management class (MGMTCLAS); all information about the signed-on user already resides in memory.

For production data sets whose High Level Qualifier does not match any defined User or Control ACID, the RESOWNER field should be added to the data set ownership element. To optimize performance, the RESOWNER should be set to the ACID which most commonly creates the data sets that begin with this High Level Qualifier.

Chapter 5: Implementing Security for IMS

This section contains the following topics:

CICS, IMS, and CA IDMS Considerations

Considerations for implementing CA Top Secret for CICS, IMS, and CA IDMS are similar because the security interfaces behave in a similar fashion. The transactions and resources available within CICS, IMS, and CA IDMS are basically the same resources available in TSO and CA Roscoe, but have different names. This table shows the relationship among the resources appropriate to each facility.

TSO/CA Roscoe	CICS	IMS	CA IDMS
Commands	Transactions	Transactions	Task Codes
Programs	PPTs	PSBs	Sub-Schemas and Programs
Data Sets	FCTs, DCTs, JCTs, TSTs	DBDs	Area

The resources in CICS, IMS, and CA IDMS that are similar to data sets are treated as different resource types by CA Top Secret. The data sets made available to the regions of the different facilities are opened directly by the regions, and not by the users signing on to those regions. The region ACID controls access to the data sets, and the user ACID will control access to the resource type appropriate to that facility–for example, FCTs within CICS.

CICS, IMS, and CA IDMS Implementation Strategy

CICS, IMS, and CA IDMS lend themselves to a gradual implementation strategy, since they usually run in multiple regions.

Plan your implementation by region, implementing CA Top Secret in the test regions first and migrating into production. You can enter your CA Top Secret definitions through TSO, CA Roscoe, or BATCH before you interface CA Top Secret with your first selected region.

The same general mode strategies apply for all of these facilities:

- Put the region in IMPLEMENT mode so that defined users are controlled through profile access definitions and undefined users are controlled through the security native to the particular facility.
- As you define new sets of users to CA Top Secret test each set by assigning WARN mode to its profile.
- Revoke WARN mode from the profile when you are satisfied that the users are properly defined. They will then be effectively treated as though they are in FAIL mode.
- When all the users are defined and selected resources protected migrate the region to FAIL mode.

The DEFPROT attribute can control whether CICS, IMS, and CA IDMS resources are protected by default or by definition in all modes. If you choose not to give default protection to them, you can include a gradual implementation strategy for resources. An effective approach is to protect users and resources by application. This approach allows the security administration group to concentrate on the requirements of each application, and to design a security scheme that is most effective for that application. Even if the region is already in FAIL mode, you can continue to protect resources by testing the definitions with ACIDs in WARN mode, and then making those resources available to the appropriate users.

CICS, IMS, and CA IDMS Native Security

Each of these facilities has its own native security that coexists with CA Top Secret in specific modes. In DORMANT mode, each facility has complete control over access through the native security available in that facility. In WARN and IMPLEMENT modes, native security and CA Top Secret can, in most cases, coexist.

CA Top Secret replaces CICS native security in FAIL mode. IMS and CA IDMS provide additional native security features; for this reason, CA Top Secret coexists with IMS and CA IDMS native security in FAIL mode, allowing further levels of control. Turn off the IMS and CA IDMS native security if this is appropriate for your organization.

IMS to CA Top Secret Definition

The IMS batch environment, BTS, and similar products are single user IMS address spaces.

The IMS/TM environment is a multi-user address space environment. It is defined to CA Top Secret through the Facilities Matrix, and may initiate in the system as a started task or batch job.

IMS batch regions are single user address spaces that do not use a CA Top Secret IMS FACILITY like IMSPROD. CA Top Secret security in this environment is controlled by the DL1B control option. If the DL1B control option is specified, CA Top Secret performs PSB and DBD security for all IMS batch jobs.

CA Top Secret can be used to protect PSBs and DBDs for IMS regions running BTS or other similar products. However, BTS edits DL1 calls and must be modified to support Application Interface calls.

Batch jobs running Batch Message Processing (BMP) regions are dependent on an associated IMS control region. The connection between the regions is established by providing a value for IMSID in the BMP IMSBATCH procedure that matches the IMSID assigned to the IMS control region. Security for the BMP environment is controlled by the security processing for the IMS control region.

IMS Multi-User Address Space Environment

Under IMS/TM, each user signed on to an IMS control region occupies part of the IMS control region address space. When the IMS user requests access to a resource, the operating system "sees" IMS performing the access-not the individual user. To protect resource accesses on the individual user level, IMS issues its own security checks on behalf of the user requesting the resource access.

Facility Matrix Resource Translation

For some IMS resources, IMS generates resource security calls using IMS defined resource classes. For example, IMS generates transaction security calls using the TIMS resource class.

CA Top Secret translates IMS generated resource class requests into traditional TSS defined resource classes. To implement resource translation, use the RXLTADD operand to add translation entries to the facility definition.

The following table shows the IMS objects you can perform resource class translation on.

Object	IMS Standard Class	IMS Generated Class	TSS Class
Transactions	TIMS	Txxx	OTRAN
Terminals	LIMS	Lxxx	TERMINAL
PSBs	IIMS	Ixxx	PSB
Databases	PIMS	Pxxx	DBD

Example: IMS translation

This example translates IMS generated TIMS transaction security requests to the traditional TSS OTRAN resource class for the IMSPROD facility:

```
TSS MODIFY FACILITY(IMSPROD=RXLTADD(TIMS:OTRAN))
```

After this command is executed security calls with the TIMS resource class are checked against OTRAN user permissions.

NOXDEF and XDEF Suboptions

The NOXDEF FACILITY suboption is set by default to allow all users to execute any unowned transaction until access controls have been established by either an inclusive or exclusive list for the user. To provide default protection for a transaction, set the XDEF FACILITY suboption. The XDEF suboption indicates that users must have some kind of transaction list, either TRANSACTIONS or XTRANS, before they can execute an unowned transaction.

Define IMS as a Started Task

To define the IMS control region to CA Top Secret as a started task, enter:

```
TSS CREATE(IMSCACID) TYPE(USER)
                     NAME('PRODUCTION CNT IMS')
                     DEPARTMENT(dept)
                     FACILITY(STC)
                     MASTFAC(IMSPROD)
                     PASSWORD(NOPW,0)
                     NOLCFCHK (NOSUBCHK]
```

Notes:

- The ACID associated with the started task must be created with FACILITY(STC) to allow execution as a started task
- The department must already exist
- If the MASTFAC keyword is not specified, the facility used defaults to IMSPROD
- The IMS control region must be defined to CA Top Secret with an ACID that has the NOLCFCHK attribute.
- If the control region uses the spool interface to submit jobs, the control region ACID must have cross-authorization for any ACID used in the submitted jobs. To bypass this requirement, specify the NOSUBCHK attribute for the control region ACID.
- Protected data sets opened by the control region must be permitted to the ACID
- Defining a started task to CA Top Secret results in the association of that STC with a specified ACID.

Example: IMS as a started task

This example associates procedure PRODIMS with Region ACID IMSCACID:

```
TSS ADDTO(STC) PROCNAME(PRODIMS)
               ACID(IMSCACID)
```

Define IMS as a Batch Job

To define the IMS control region to CA Top Secret as a batch job, enter:

```
TSS CREATE(IMSCACID) TYPE(USER)
                     NAME('PRODUCTION CNT IMS')
                     DEPARTMENT(dept)
                     FACILITY(BATCH)
                     MASTFAC(IMSPROD)
                     PASSWORD(pwd)
                     NOLCFCHK (NOSUBCHK]
```

Notes:

- The ACID is created with FACILITY(BATCH) to allow execution as a batch job
- The department must already exist
- When users sign on to the control region, they must be associated with the same facility as the MASTFAC associated with the ACID of the control region.
- If the MASTFAC keyword is not specified, the facility used defaults to IMSPROD.
- Protected data sets and other resources opened by the control region must be permitted to the ACID.
- The IMS control region must be defined to CA Top Secret with an ACID that has the NOLCFCHK attribute.
- If the control region uses the spool interface to submit jobs, the control region ACID must have cross-authorization for any ACID used in the submitted jobs. To bypass this requirement, specify the NOSUBCHK attribute for the control region ACID.
- Code the USER=acid keyword on the batch job card, then submit the necessary JCL. Using a password is desirable for batch submission of a control region.

Define IMS Message Region as a Started Task

To define the IMS message region to CA Top Secret as a started task, enter:

```
TSS CREATE(IMSMACID) TYPE(USER)
                     NAME('PRODUCTION MSG IMS')
                     DEPARTMENT(dept)
                     FACILITY(STC)
                     MASTFAC(imsfac)
                     PASSWORD(NOPW) (NOSUBCHK]
```

Notes:

- The ACID that is associated with the started task must be created with FACILITY(STC) to allow execution as a started task.
- The department must already exist.
- Because some IMS security processes may occur in the IMS message processing region, the MASTFAC keyword must be specified, and the facility “imsfac” must match the MASTFAC of the IMS control region.
- If the message processing region uses the spool interface to submit jobs, the message processing region ACID must have cross-authorization for any ACID used in the submitted jobs. To bypass this requirement, specify the NOSUBCHK attribute for the message processing region ACID.

Defining a started task to CA Top Secret results in the association of that STC with a specific ACID.

```
TSS ADDTO(STC) PROCNAME(IMSMPR)
               ACID(IMSMACID)
```

Define the IMS Message Region as a Batch Job

To define the IMS message region to CA Top Secret as a batch job, enter:

```
TSS CREATE(IMSMACID) TYPE(USER)
                     NAME('PRODUCTION MSG IMS')
                     DEPARTMENT(dept)
                     FACILITY(BATCH)
                     MASTFAC(imsfac)
                     PASSWORD(pwd) (NOSUBCHK]
```

Notes:

- The ACID is created with FACILITY(BATCH) to allow execution as a batch job.
- The department must already exist.
- Because some IMS security processes may occur in the IMS message processing region, the MASTFAC keyword must be specified, and the facility “imsfac” must match the MASTFAC of the IMS control region.
- If the message processing region uses the spool interface to submit jobs, the message processing region ACID must have cross-authorization for any ACID used in the submitted jobs. To bypass this requirement, specify the NOSUBCHK attribute for the message processing region ACID.
- Code the USER=acid keyword on the batch job card, then submit the necessary JCL. A password may be desirable for batch submission of a message region.

Define a BMP Region as a Started Task

To define a BMP region to CA Top Secret as a started task by creating a started task ACID, enter:

```
TSS CREATE(bmpregn) TYPE(USER)
                    NAME(bmp-region-name)
                    DEPT(department)
                    FAC(STC,imsfac)
                    MASTFAC(imsfac)
```

To define the ACID with a started task procedure name, enter:

```
TSS ADD(STC) PROCNAME(stcname)
             ACID(bmpregn)
```

Because some IMS security processes may occur in the IMS BMP region, the MASTFAC keyword must be specified, and the facility “imsfac” must match the MASTFAC of the IMS control region.

Define a BMP Region as a Batch Job

To define a BMP region to CA Top Secret as a batch job, create a JOB ACID and employ the acid on the JOB statement of the JCL which invokes the BMP region:

```
TSS CREATE(bmpregn) TYPE(USER)
                    NAME(bmp-region-name)
                    DEPT(deparment)
                    FAC(BATCH,imsfac)
                    MASTFAC(imsfac)
```

Because some IMS security processes may occur in the IMS BMP region, the MASTFAC keyword must be specified, and the facility “imsfac” must match the MASTFAC of the IMS control region.

The MRO Option

Under normal operation, CA Top Secret for IMS shares a signon XREF table in ECSA between the IMS control region and its associated message regions. This implementation can reduce the number of signons performed by an IMS subsystem and reduce the number of consequent I/O's to the security file when PSB, DBD, or DB2 security is involved. In situations where only transaction security takes place, the ECSA table is unlikely to provide any efficiency while adding additional ECSA requirements for the common XREF table. The ECSA XREF table has no effect on IMS MSC security as implemented in CA Top Secret.

For situations where ECSA is at a premium, the control option OPTIONS(19) can be added to the CA Top Secret PARMLIB to disable the use of a common XREF table between the IMS control region and its associated messages region. When OPTIONS(19) is in force, the XREF tables for the control region and message regions are normally kept separately in private storage. Exceptions can be arranged for particularly intensive IMS systems when OPTIONS(19) is in force, to allow such systems to operate more efficiently with the ECSA XREF table.

To add the MRO option to the control region acid and to each of the associated message regions, enter:

```
TSS ADDTO(control region acid) MRO
```

```
TSS ADDTO(message regions acid) MRO
```

Failure to add the MRO attribute to all associated regions can cause abnormal termination of IMS. The MRO attribute has no effect if OPTIONS(19) is disabled, which is the default.

Control Options for IMS

The DL1B and IMS control options are specific to the IMS facility. For information, see the *Control Options Guide*.

Modes of Operation

The following table summarizes access operation by mode.

FUNCTION	DORM MODE	WARN MODE	IMPL MODE	FAIL MODE
Signon Processing Required		X	X	X
Full Password Controls	1	2	2	X
Transaction Security Active		3	X	X
Command Security		3	X	X
RAS Security (IMS r9.1 and above)		3	X	X
PSB Protection Active (BMPs, MPPs)		3	2	X
PSB Protection Active (DLIBATCH)		3	5	6
DBD Protection Active (BMPs, MPPs)		3	2	X
DBD Protection Active (DLIBATCH)		3	5	6
IMS LOCK Command Resource Security (IMS 9.1 and above)		3	X	X
OTMA Resume TPIPE Security (IMS r9.1 and below)		3	X	X
Terminal Protection Active		3	X	X
Signon User Exit Invoked (DFSCSGN0)	X	X	4	X
Transaction User Exit Invoked (DFSCTRN0)	X	X	X	X
TSS Command Available	X	X	X	X
Application Interface Available	X	X	X	X
Logging Available		X	X	X
Auditing Available		X	X	X

X

All Users in All Conditions

1

If Explicit Signon is Performed

2

For Defined Users Only

3

Messages and Logging Only

4

For Undefined Users Only

5

If DL1B is YES, then security is enforced for defined users. If set to NO (the default), then no security takes place.

6

If DL1B is YES, then security is enforced for all users. If set to NO (the default), then no security takes place.

Note: Security Control Administrators-that is, MSCAs, SCAs, LSCAs, ZCAs, VCAs, DCAs-can use their CA Top Secret password in any mode.

Violation Thresholds

Terminal locking due to excessive violations is enforced through CA Top Secret. If the VTHRESH control option is in effect and the action is either CAN (for cancel) or SUS (for suspend), CA Top Secret locks the terminal when the threshold is exceeded. Your only course of action is to sign off. If the action is VTHRESH(SUS), you are automatically suspended and will not be able to sign on until unsuspended by a security administrator.

TYPE=IMS Facility Control Options and Behaviors

Some facility sub-options behave differently in IMS than they do in other facility types.

SIGN(M|S)

Depending upon the startup parameters set in the IMS control region, users may be restricted to signon only at a single static terminal. In this case, setting a TSS facility SIGN(M) sub-option to allow a single ACID to signon at multiple IMS terminals has no effect.

ETO terminals which employ a DFSUSER descriptor at signon are automatically restricted to single signons. Initialization parameters and customization exits for IMS can be used to alter the signon descriptor, and allow multiple signons for ETO terminals. However, some IMS application op-codes (for example CHNG) perform security INIT using the descriptor as an ACID. In these cases, the administrator is responsible to assure that such ACIDs exist, and are capable of signon to the associated region. SIGN(M) may be used in such cases.

LUMSG\NOLUMSG and STMSG\NOSTMSG

Normally, LUMSG and STMSG govern the issuance of the messages TSS7001I and TSS7000I respectively.

When only one of these facility options is enabled, only one line of message is generated for a successful signon. This has the effect of releasing a successful signon message in format DFSMO1 rather than the usual DFSMO2.

When both of these options are disabled, the response to a successful signon is to release message DFS3650I on format DFSMO2, displaying both user and node information. This occurs whether the end user is signed onto a static terminal or an ETO terminal.

Define CA Top Secret to IMS

To define IMS to your CA Top Secret system:

- Generate the IMS external security interface
- Install the CA Top Secret transaction
- Enable the sign on screen
- Set IMS control region initialization

Generate the IMS External Security Interface

The SECURITY macro statement in the IMS system definition lets you specify security features in effect for the IMS system being defined. These security features can be overridden during IMS system initialization by IMS initialization parameters.

The following parameters for the SECURITY macro are used to select CA Top Secret for IMS security*:*

```
SECURITY SECLVL=(TRANAUTH,SIGNON),
          TYPE=(RACFTERM,RACFCOM,RACFAGN|RASRACF),
       . . .
```

TRANAUTH

Specifies that transaction authorization is to be performed.

SIGNON

Specifies that user ID verification is to be performed.

RACFTERM

Specifies that CA Top Secret will be used for transaction authorization.

RACFCOM

Specifies that CA Top Secret will be used for command authorization.

RACFAGN

Specifies that CA Top Secret will be used for application group name (AGN) authorization during IMS dependent region initialization.

RACFAGN is valid in IMS r9.1 and below, and is mutually exclusive with the RASRACF option.

RASRACF

Specifies that CA Top Secret will be used for resource access security authorization during IMS dependent region processing. RASRACF is valid in IMS r9.1 and above, and is mutually exclusive with the RACFAGN option.

The other parameters for the SECURITY macro should be specified as appropriate for your installation.

Note: The RCLASS keyword in the SECURITY macro should normally not be used. CA Top Secret is designed to look for resources using the default value (RCLASS=IMS) in the resource class names, for example, TIMS for transactions and CIMS for commands. If a site chooses a non-default RCLASS, they are responsible for defining the resulting resource classes. Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate FACILITYs for distinguished regions and to distinguish region-specific permissions by FACILITY.

Install TSS Transaction

To enable the TSS transaction, code the following APPLCTN macro in your IMS system definition:

```
APPLCTN PSB=TSSIMS,
PGMTYPE=(TP),
SCHDTYP=PARALLEL
TRANSACT CODE=TSS,
PRTY=(n,n,n),
INQUIRY=YES,
MODE=SNGL
```

Note: The TRANSACT CODE=TSS cannot be customized.

Copy the CA Top Secret for IMS transaction load module TSSIMS from the CA Top Secret CAILIB into the IMS PGMLIB program library.

Perform a PSB gen and an ACB gen for the CA Top Secret TSSIMS PSB. Only the following control card is required to perform the PSB gen:

```
PSBGEN PSBNAME=TSSIMS,LANG=ASSEM
```

Perform the IMS System Definition

To implement the IMS system definition changes, perform an IMS system definition at the NUCLEUS level or higher. For information, see IBM's *IMS Installation Volume 2: System Definition and Tailoring*.

Enable the Signon Screen (Optional)

DFSMO1 and DFSMO2 are normally employed for static terminal signon to pass online CA Top Secret messages back to the end user. These MFS format members are supplied by IMS during Stage 2 IMS Generation. Normally, the default DFSMO1 is deployed for single line signon message returns (usually errors). DFSMO2 is deployed for multiple line signon message returns (usually a success).

For ETO terminals, format DFSIGNP is presented when an un-generated terminal connects to the IMS region. DFSIGNP is created during Stage 2 IMS generation. DFSIGNP, DFSMO1, and DFSMO2 may be used to pass messages and return codes back to the IMS terminal user.

When an ETO terminal is connected to IMS before successful signon:

- It can only issue the commands:

  ```
  /SIGN
  /RCL
  ```

- It cannot execute transactions

The format DFSIGNP:

- Generates the implicit command:

  ```
  /SIGN ON
  ```

- Processes the command normally when the screen is entered

You can customize these "signon" format members, but must conform to IMS specifications. A customized version of DFSMO1 is supplied in the TSSOPMAT library. Customization of IMS signon formats is the responsibility of the administrator. For information, see the documents for the IMS Message Formatting Services supplied for your current release of IMS.

IMS Control Region Initialization Requirements

IMS Control Region Initialization Requirements The following IMS initialization parameters are used to control CA Top Secret in the IMS control region. These initialization parameters can be specified in the IMS procedure JCL or in the DFSPBxxx parameter block member in the IMS PROCLIB*:*

RCF=A|B|C|N|R|S|T|Y (IMS r9.1 and below)

RCF=A|C|N|S|T|Y (IMS r10 and above)

Controls whether CA Top Secret will be used for signon, transaction, and command authorization.

A

Enables CA Top Secret for signon, transactions, and commands. Combines options S, T, and C.

B

(IMS r9.1 and below) Enables CA Top Secret for signon, transactions, and commands (combines options S, T, and C), but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set.

C

Enables CA Top Secret for commands from ETO terminals.

N

Specifies that CA Top Secret will not be used for signon, transaction, and command authorization.

R

(IMS r9.1 and below) Enables CA Top Secret for commands from static and dynamic (ETO) terminals (option S), but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set.

S

Enables CA Top Secret for signon and for commands from static and dynamic (ETO) terminals.

T

Enables CA Top Secret for signon and transaction authorization.

Y

Enables CA Top Secret for signon and transaction authorization, and for commands from ETO terminals, but not for commands from static terminals. (Combines options T and C)

SGN=D|E|F|G|M|N|W|X|Y|Z (IMS r9.1 and below)

SGN=F|G|M|N|Y|Z (IMS r10 and above)

Controls whether signon verification is active and whether the IMS signon verification security table is loaded from IMS MATRIX data set.

D

(IMS r9.1 and below) Enables signon verification and the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set. The operator cannot override this option on the restart command during IMS initialization.

E

(IMS r9.1 and below) Enables signon verification, but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set. A single user is allowed to sign on to multiple IMS terminals. The operator cannot override this option on the restart command during IMS initialization. (Combines options D and M)

F

Enables signon verification. The operator cannot override this option on the restart command during IMS initialization.

G

Enables signon verification and allows a single user to sign on to multiple IMS terminals This option cannot be overridden by the operator on the restart command during IMS initialization. (Combines options F and M).

M

Enables signon verification and allows a single user to sign on to multiple IMS terminals.

N

Signon verification is not enabled. For r9.1 and below the IMS signon verification security table (DFSISSOx) is not loaded from the IMS MATRIX data set. The operator can override this option on the restart command during IMS initialization.

W

(IMS r9.1 and below) Enables signon verification, but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set. The operator can override this option on the restart command during IMS initialization.

X

(IMS r9.1 and below) Enables signon verification, but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set. A single user can sign on to multiple IMS terminals. The operator can override this option on the restart command during IMS initialization. (Combines options W and M)

Y

Enables signon verification. For IMS r9.1 and below the IMS signon verification security table (DFSISSOx) is loaded from the IMS MATRIX data set. The operator can override this option on the restart command during IMS initialization.

Z

Enables signon verification and allows a single user to sign on to multiple IMS terminals. For r9.1 and below the IMS signon verification security table (DFSISSOx) is loaded from the IMS MATRIX data set. The operator can override this option on the restart command during IMS initialization. (Combines options Y and M).

- **Notes:**
- IMS multiple signon support (SGN=E|G|M|X|Z) applies only to static IMS terminals and is not supported for ETO terminals.
- When IMS multiple signon is enabled (SGN=E|G|M|X|Z), single signon can be enforced through the CA Top Secret FACILITY option SIGN(S). When specified in this way, multiple attempts to sign on are audited by CA Top Secret and error messages are generated if a user tries to sign on multiple times.

TRN=E|F|N|X|Y (IMS r9.1 and below)

TRN=F|N||Y (IMS r10 and above)

Controls whether transaction security authorization is active and whether the IMS signon verification security table is loaded from IMS MATRIX data set.

E

(IMS r9.1 and below) Transaction authorization checking is enabled, but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set. The operator cannot override this option on the restart command during IMS initialization.

F

Transaction authorization checking is enabled. The operator cannot override this option on the restart command during IMS initialization.

N

Transaction authorization checking is not enabled. The operator can override this option on the restart command during IMS initialization.

X

(IMS r9.1 and below) Transaction authorization checking is enabled, but the IMS signon verification security table (DFSISSOx) will not be loaded from the IMS MATRIX data set The operator can override this option on the restart command during IMS initialization.

Y

Transaction authorization checking is enabled. The operator can override this option on the restart command during IMS initialization.

AOIS=A|C|N|R|S

Controls security for type 2 AOI commands, that is, commands issued by applications programs using the ICMD DLI communications call.

A

Enables CA Top Secret security for type 2 AOI commands, and indicates that the optional IMS DFSCCMD0 command security user exit should also be called. (Combines options C and R)

C

Specifies that the optional IMS DFSCCMD0 command security user exit should be called for command authorization.

N

Specifies that no ICMD communications call can be issued by applications programs.

R

Specifies that CA Top Secret security will be used for type 2 AOI commands.

S

Specifies that no security will be used for type 2 AOI commands.

AOI1=A|C|N|R|S (IMS r9.1)

AOI1=A|C|N|R (IMS r10 and above)

Controls security for type 1 AOI commands (commands issued using the CMD DLI communications call).

A

Enables CA Top Secret for type 1 AOI commands, and indicates that the optional IMS DFSCCMD0 command security user exit should also be called. (Combines options C and R)

C

Specifies that the optional IMS DFSCCMD0 command security user exit should be called for command authorization.

N

Specifies that no CMD communications call can be issued by applications programs.

R

Specifies that CA Top Secret will be used for type 1 AOI commands.

S

(IMS r9.1 only) Specifies that IMS internal security is used for type 1 AOI commands.

Note: If AOI1=A, C, or R is specified, the AOI parameter must be specified in the TRANSACT macro in the IMS system definition for any transactions that issue the CMD DLI communications call. For information, the IBM manual *IMS Installation Volume 2: System Definition and Tailoring*.

ISIS=0|1|2 (IMS r8.1 and below)

ISIS=0|1|2|A|C|N|R (IMS r9.1)

ISIS=A|C|N|R (IMS r10 and above)

Controls AGN (application group name) security for IMS r9.1 and below, and controls RAS (resource access security) for IMS r9.1 and above. AGN and RAS security are mutually exclusive and cannot both be specified in IMS r9.1.

0

(IMS r9.1 and below)) No AGN security is performed. If this option is specified, CA Top Secret for PSBs and DBDs and the use of the TSS command and the Application Interface under IMS are disabled

1

(IMS r9.1 and below) Enables CA Top Secret for AGNs

2

(IMS r9.1 and below) CA Top Secret for AGNs is disabled, but the optional IMS DFSISIS0 user exit is called for AGN security.

A

(IMS r9.1 and above) Enables CA Top Secret RAS (resource access security) for transaction, PSB, and LTERM access by IMS dependent regions. The optional IMS DFSRAS00 resource access security user exit is also called. (combines options C and R)

C

Specifies that the optional IMS DFSRAS00 resource access security user exit should be called for authorization for transaction, PSB, and LTERM access by IMS dependent regions.

N

No RAS security is performed. If this option is specified, CA Top Secret for PSBs and DBDs and the use of the TSS command and the Application Interface under IMS are disabled

R

Enables CA Top Secret RAS (resource access security) for transaction, PSB, and LTERM access by IMS dependent regions.

Note: CA Top Secret IMS interface already provides security for PSB usage using the IMSPSBVL setting for the CA Top Secret IMS control option. For IMS r9.1 and above, if RAS PSB security has been enabled, you should specify NOIMSPSBVL for the IMS control option. Failure to do so will cause two validations to be done for each PSB, one for the RAS PSB control and one for the IMSPSBVL control.

APPCSE=C|F|N

Controls security for commands and transactions entering IMS from an APPC conversation.

C

Enables CA Top Secret for IMS commands and transactions from APPC.

F

Enables CA Top Secret for IMS commands and transactions from APPC. The security environment (ACEE) for the APPC user will be created in the dependent region when an APPC transaction is executed.

N

Disables CA Top Secret for IMS commands and transactions from APPC.

OTMASE=C|F|N

Controls security for commands and transactions entering IMS from an OTMA clients.

C

Enables CA Top Secret for IMS commands and transactions from OTMA.

F

Enables CA Top Secret for IMS commands and transactions from OTMA. The security environment (ACEE) for the OTMA user will be created in the dependent region when an OTMA transaction is executed.

N

Disables CA Top Secret for IMS commands and transactions from OTMA.

CMDMCS=B|C|N|R|Y

Controls security for IMS commands entered from an MCS or EMCS console using the command recognition character defined in the IMS system definition.

B

Enables CA Top Secret for IMS commands entered from the MCS or EMCS console using the command recognition character, and IMS will call the optional IMS DSFCCMD0 command security user exit. (Combines options C and R)

C

CA Top Secret is not called for IMS commands entered from the MCS or EMCS console using the command recognition character, but IMS will call the optional IMS DSFCCMD0 command security user exit.

N

IMS commands cannot be entered from the MCS or EMCS console using the command recognition character.

R

Enables CA Top Secret for IMS commands entered from the MCS or EMCS console using the command recognition character.

Y

IMS commands can be entered from an MCS or EMCS console using the command recognition character. No security authorization will be performed.

LOCKSEC=Y|N (IMS r9.1 and above)

Controls CA Top Secret for transactions, programs, LTERMs, and databases specified on the IMS LOCK and UNLOCK commands.

Y

Enables CA Top Secret for transactions, programs, LTERMs, and databases specified on the IMS LOCK and UNLOCK commands.

N

CA Top Secret will not be called for transactions, programs, LTERMs, and databases specified on the IMS LOCK and UNLOCK commands.

Note: The IMS LOCKSEC initialization parameter controls resource security for the IMS LOCK and UNLOCK commands. It has no effect on CA Top Secret locking with ACID LTIME, FACILITY LOCKTIME, or explicit user of the TSS LOCK command.

TCORACF=Y|N (IMS r9.1 and above)

Controls security processing for commands in TCO (time controlled operations) scripts.

Y

Enables CA Top Secret for IMS commands entered from a TCO script. IMS will also call the optional IMS DSFCCMD0 command security user exit.

N

CA Top Secret will not be called for IMS commands entered from a TCO script. IMS will call the optional IMS DSFCCMD0 command security user exit.

MSCSEC= (IMS r9.1 and above)

Controls security processing for transactions entering IMS from MSC network links

LRDIRECT

Enables CA Top Secret for MSC directed routing transactions.

LRNONDR

Enables CA Top Secret for MSC non-directed routing transactions.

LRALL

Enables CA Top Secret for all MSC transactions, both directed routing transactions and non-directed routing transactions.

LRNONE

Disables CA Top Secret for all MSC transactions.

Note: CA Top Secret IMS interface already provides security for MSC non-directed routing transactions using the IMSMSC field in the IMS control region ACID. If the IMSMSC field is being used to control security for MSC non-directed routing transactions, do not specify LRNONDR or LRALL for the MSCSEC parameter. Doing so will cause two validations to be done for each transaction, one for the IMSMSC control and one for the MSCSEC control.

RCFTCB=nn

This is a performance option. It specifies a number between 1 and 20 indicating how many tasks will be created in IMS to process CA Top Secret signons and signoffs.

RCVY=Y|N

This security option enables RACF password reverification during transaction and command security validation. It has no effect on CA Top Secret password reverification, which is supported as part of the normal CA Top Secret transaction and command security process.

Signon Security

CA Top Secret requires that you identify yourself to CA Top Secret to use IMS/TM. This is done by a user-initiated signon or an automatic terminal signon.

Signon Security Implementation

CA Top Secret support for user-initiated signon is controlled by the SECLVL=SIGNON parameter of the IMS system definition SECURITY macro, and by the SGN= IMS initialization parameter.

/SIGN Command

Most users have to explicitly sign on to IMS using the IMS /SIGN command.

This command has the format:

```
/SIGN acidname password [NEWPW new-pswd]
                        [VERIFY ver-pswd]
                        [GROUP group id]
                        [ACCT acct-data]
```

acidname

The user's ACID. (Required)

Range: one- to eight-characters

Password

The user's password; always required unless the ACID has a password of NOPW.

NEWPW

(Optional) Indicates that the next parameter is a new password.

new-pswd

Specifies a new password. Specifying RANDOM causes CA Top Secret to automatically generate a new random password for the user.

VERIFY

(Optional) Indicates that the next parameter is a verification password, i.e. the new password entered again to ensure accuracy**.**

ver-pswd

Specifies the new password again to ensure accuracy. IMS verifies that the new password and the verification password are the same before processing the signon request.

GROUP

(Optional) Indicates that the next parameter is a group. For information on monitoring entry by GROUP see the *User Guide*.

group id

Specifies a group ID.

ACCT

(Optional) Indicates that the next parameter is user accounting data.

Note: If used, this must be the last item in the data string.

acct-data

Specifies installation-defined accounting data to be passed to the IMS Signon Exit (DFSCSGN0). CA Top Secret does not use this data in any way.

Note: If you reassemble the optional DFSMO1 IMS default format into your MFS FORMAT library, DFSMO1 automatically formats the /SIGN command for static terminals. Format DFSIGNP may be customized for sign on at dynamic (ETO) terminals.

When a user signs on to IMS, the following security checking is performed:

- ACID identification. You must identify yourself to CA Top Secret by specifying your ACID.
- Password checking. You must specify the password for the ACID.
- New password request. You may be required to enter a new password, based on the password change interval specified for the user ACID. Other new password rules based on the NEWPW control option, such as minimum length, may also apply.
- Facility (region) checking. You must be explicitly allowed to access the facility associated with the IMS region. As delivered, CA Top Secret has two facilities for IMS: IMSPROD and IMSTEST. Additional IMS facilities may be added by the installation. (See the chapter “Defining IMS to CA Top Secret” for more details on defining a new facility.)
- **Note:** When users sign on to a control region, they must be permitted to the same facility as the MASTFAC associated with the ACID of the control region.
  ```
  TSS ADDTO(USER01) FACILITY(IMSPROD)
  ```
- Terminal security. If the terminal is owned and therefore protected, the user must be permitted to access the terminal.
- Source security. You may be restricted to signing on to a terminal, or a group of terminals.
- Duplicate signon. You may be restricted from signing on to more than one terminal at a time with the same ACID.
- Suspended or expired. You will not be allowed to sign on if your ACID has a suspended or expired password.
- Last-used information and messages If all the above checks are successful, the user will be allowed to sign onto IMS. When the facility suboption LUMSG is in force, the TSS7000I last-used message is issued. To minimize security file I/O, NOLUUPD and NOLUMSG may be set in the facility; this suppresses the update and display of last-used information for signons in the IMS facility.
- Status messages. The TSS7001I status message accompany the last-used message.

Automatic Terminal Signon

CA Top Secret allows for the automatic signon of an IMS terminal. This is useful for applications requiring minimal security, or where the physical security surrounding a terminal is adequate for the sensitivity of the application. Automatic signon is also useful for receive-only terminals, used for displaying secured information.

Automatic terminal signon is performed whenever a transaction is entered prior to performing an IMS signon, and there exists a CA Top Secret ACID with the same name as the IMS terminal.

The terminal name used depends on your environment and control option settings:

- VTAM terminals use the VTAM node name
- BTAM or LU6.2 terminals use IMS LTERM

Terminals with ACIDs defined automatically become eligible for automatic terminal signon. However, the ATS ACID must be:

- Allowed to the IMS facility
- PERMITted for the terminal, if the terminal is an owned resource

If there is no ACID defined with the same name as the terminal defined to CA Top Secret, the control option IMS is checked. If IMS(NOIMSATSDF) is specified, the transaction fails. If IMS(IMSATSDF) is specified, the FACILITY DEFACID is checked. If DEFACID(NONE) is specified, the transaction is failed and the user receives a message requesting that an explicit signon be performed. However, if the DEFACID exists, then CA Top Secret performs the security checks (4) through (8) as previously described for security checking. If these checks are successful, the ACID becomes associated with that terminal until signoff just as if an explicit signon had been performed, and processing of the transaction that was entered is initiated (subject to LCF transaction security).

Notes

- If the automatic terminal signon is successful, the user who entered the transaction will not be aware of the process.
- ETO terminals cannot use Automatic Terminal Signon since IMS will not allow users to enter a transaction or command until after an explicit signon has been done.

Configure Automatic Terminal Login

Security administrators can configure automatic terminal login on fixed terminals to avoid unnecessary user logins.

Note: If an ACID exists with the same ID as the PTERM and a transaction ID is entered at the terminal, CA Top Secret logs in automatically as the terminal ACID before the transaction is executed.

Follow these steps:

1. Code the terminals into the region's Stage 1 macros. For example:

```
   TYPE    UNITYPE=(3270,LOCAL),
                MODEL=2,
                FEAT=(PFK,NOCD),
                UNIT=3277,
                PTRSIZE=IGNORE
    SPACE 2
    :::::
    :::::
    :::::
    SPACE 2
 ** FIXED TERMINALS
 A61L0901 TERMINAL  NAME=A61L0901
          NAME L61L0901
 A61L0902 TERMINAL  NAME=A61L0902
          NAME L61L0902
 A61L0903 TERMINAL  NAME=A61L0903
          NAME L61L0903
    :::::
    :::::
    :::::
```

   The Stage 2 gen creates terminal descriptors in the PROCLIB member DFSDSCMx. The terminal descriptors are generated as comments in the PROCLIB member. For example:

```
* U A61L0901 LTERM=L61L0901
* U A61L0902 LTERM=L61L0902
* U A61L0903 LTERM=L61L0903
* U A61L0904 LTERM=L61L0904
* U A61L0905 LTERM=L61L0905
```

   Note: If the static terminals are left as comments, the descriptor for the terminal defaults to ETO DFSUSER, forcing the user to log in to access the terminal.

2. Uncomment a terminal descriptor to enable automatic login on that terminal.

IMS Resource Security

CA Top Secret provides security for the IMS resources:

- Transactions
- IMS commands
- Application Group Names (AGNs) in IMS r9.1 and below
- Resource Access Security (RAS) resources in IMS r9.1 and above
- PSBs
- DBDs
- Resources on the LOCK and UNLOCK commands in IMS r9.1 and above
- OTMA TPIPE names on the Resume TPIPE request in IMS r10 and above

For these resources to be protected, they must be owned. The standard set of CA Top Secret control features for owned resources is also available for IMS resources, including:

- Generic prefixing
- Expiration, day of week, and time of day limiting
- Facility (region) restriction
- Action control (specifies which TSS action will take effect if rules in the PERMIT function are enforced)
- Program pathing
- I/O access levels
- ACTIONS

An ownable resource can be added to the AUDIT record, so that all activity involving it can be tracked. Resource ownership allows more comprehensive, flexible, and systematic security than a transaction-based security.

Note: The owner of an IMS resource should be a non-USER type ACID. When ownership is added in this way, the owner cannot sign on to use the resources defined, since TYPE=DEPT and other organizational type users are not allowed to sign on. However, if ownership is granted to an ACID with TYPE=USER, that user automatically has complete access to the resources which cannot be overridden by the commands:

```
TSS PERMIT ..... ACTION(DENY)
```

```
TSS PERMIT ..... ACCESS(NONE)
```

Transaction Security

Transactions can enter IMS from:

- An IMS terminal
- An application program using a CHNG or ISRT DLI communication call
- Another IMS region over an MSC network link
- An APPC conversation
- An OTMA client
- From an OM environment using the OM QUE TRAN command

OTRAN Security

The OTRAN resource name is shared by all IMS, CICS, and CA IDMS® facilities. Protecting a transaction via OTRAN for an IMS region results in transactions of the same name being protected in all IMS, CICS, and CA IDMS regions that are also under the control of CA Top Secret.

Notes:

- A transaction protected via OTRAN will not go through LCF checking.
- Although OTRAN may be set with the MASK RDT attribute, this is not supported in the IMS environment. Clients employing TSS security in IMS are cautioned not to set the MASK attribute in the OTRAN RDT definition.

To add ownership of a transaction to an ACID, enter:

```
TSS ADDTO(acid) OTRAN(transaction)
```

To allow users access to the transaction, enter:

```
TSS PERMIT(acid) OTRAN(transaction)
```

LCF Security

If you choose not to protect transactions using OTRAN, they can be protected using LCF. Transactions protected through LCF must be defined by facility. Transactions can be defined either inclusively (TRANS) or exclusively (XTRANS), but not both. Essentially, each user can have an inclusive list, which specifies a list of transactions the user is allowed, or an exclusive list that the user is not allowed to use. Transactions should be divided by function or subset and defined as a group within profiles. Transactions are defined only once per group, instead of once per user.

SAF Security

For many of the ways that transactions enter an IMS system, IMS uses a SAF call to invoke CA Top Secret transaction security. The resource class for these transaction security calls is formed from the prefix "T" and the value established for the RCLASS parameter (which defaults to "IMS").

Note: The OPTIONS(27) control option can be turned on in order for the IMS SAF based TIMS resource class checks to be translated to LCF and OTRAN as needed (rather than using TIMS).

CA Top Secret provides a system-supplied resource class TIMS.

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

If the administrator chooses to use a non-default RCLASS value, the administrator is responsible for:

- Creating an RDT entry for the resulting transaction resource class, concatenating "T" with the RCLASS value. Attributes of the new class should be identical with TIMS.
- Informing support of the non-standard usage if problems are encountered.

The following instructions assume the use of the TIMS facility for transaction security. The administrator should substitute their non-standard transaction resource class, if one is in use.

TIMS is a general resource that can be ADDed to establish ownership:

```
TSS ADDTO(acid) TIMS(transaction)
```

To allow users access to the transaction, enter:

```
TSS PERMIT(acid) TIMS(transaction)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) TIMS(transaction)
```

The first permission allows the user to execute the transaction only in regions using the IMSPROD facility. The second permission allows the user to execute the transaction unrestricted by facility.

Transaction Security Implementation

CA Top Secret transaction security is controlled by the SECLVL=TRANAUTH and TYPE=RACFTERM parameters of the IMS system definition SECURITY macro, and by the RCF= and TRN= IMS initialization parameters.

Transactions from Terminals

Transaction validation is performed for transactions entered directly from an IMS terminal.

Note: Transaction validation is also performed during the validation of the IMS operator command:

```
/SET TRANSACTION transaction
```

CA Top Secret secures terminal transactions using either OTRAN (resource) security or the Limited Command Facility (LCF).

Transactions from Application Programs

IMS application programs can initiate IMS transactions by performing a program-to-program message switch, that is, the application program issues an ISRT DLI communications call to a non-modifiable TP PCB that specifies a transaction destination, or the application program issues a CHNG DLI communications call to a modifiable TP PCB that specifies a transaction destination, followed by ISRT calls to the new destination.

For program-to-program message switches that use an ISRT DLI communications call to a non-modifiable TP PCB, no CA Top Secret transaction validation is performed.

For program-to-program message switches that use a CHNG DLI communications call to a modifiable TP PCB, a SAF call is performed that provides the CA Top Secret transaction validation in the TIMS resource class. See the SAF Security section in this chapter, for more information on the SAF calls for transaction validation.

Note: When the application program issuing the CHNG call is a BMP, the ACID used for the transaction validation depends on the value of the BMPUSID parameter specified in IMS DFSDCxxx data communications PROCLIB member:

- When BMPUSID=USERID is specified, a signon is performed for the ACID assigned to the BMP job or task and this ACID is used for the transaction validation. If the ACID is not allowed to signon to the control region, CHNG call transaction security is ignored.
- When BMPUSID=PSBNAME is specified, a signon is performed for an ACID with the same name as the PSBNAME specified in the BMP JCL (not the PSBNAME for the transaction being inserted), and this ACID is used for the transaction validation. If the ACID does not exist or if the ACID is not allowed to signon to the control region, CHNG call transaction security is ignored.

Transactions from MSC

Transactions can enter an IMS region from another IMS system over an MSC network link. These transactions can be divided into two types:

- Directed routing transactions might not be defined in the sending IMS system. An application program initiates a directed routing transaction by specifying the transaction code and the ID of the target IMS region to which the transaction should be sent.
- Non-directed routing transactions are defined in the sending IMS system. An application program initiates a non-directed routing transaction by specifying the transaction code. The transaction definition in the sending IMS system includes the ID of the target IMS region to which the transaction should be sent.

For directed routing transactions, no CA Top Secret transaction validation is performed in the receiving IMS system.

For non-directed routing transactions, CA Top Secret allows you to perform a transaction validation and to choose what userid should be used for validation process.

Transaction validation for MSC non-directed routing transactions is determined by the values of the IMSMSC field of the IMS control region ACID of the receiving system.

The format to specify the IMSMSC values is:

```
TSS ADDTO(acid) IMSMSC('msname1(option),msname2(option)...')
```

The *acid* value is the ACID for the IMS control region of the receiving IMS system.

The msname operands identify the MSC links in the receiving IMS system the options are defined for. The values for the msname operands are the labels on the MSNAME statements that define the MSC logical links as defined in the receiving system IMS system definition. Use the linkname ALL for all previously unspecified linknames in the Stage 1 gen for this region to receive the associated link option.

For each MSC link, the option value determines the ACID used for the transaction validations:

acid

Specifies the default ACID for this MSC link. This ACID is signed on during IMS control region initialization and used for the validation of all non-directed routing transactions received on the MSC link. Ensure that the acid specified is not used for session sign on (for example through SOURCE restriction). An ACID used for an IMSMSC link and online session sign on has unpredictable effects.

+USER

Specifies that the userid signed on in the originating IMS region is propagated to the MSC receiving region. It is the administrator's responsibility to assure that the session user is allowed to sign on in both the originating and receiving regions and that the transaction is allowed in both regions.

+DEFAULT

Specifies that the DEFACID assigned by the receiving region FACILITY control option is used for MSC transaction validation for the MSC linkname.

For performance, specifying a single ACID for all in-bound transactions may be the best choice. If using PSB security, DBD security, or the application interface for MSC routed transactions, specifying +USER is a better choice. PSB, DBD, and application interface calls are message region security events and are always performed against the userid contained in the transaction's IOPCB. Specifying +USER provides consistent security between MSC transaction verification and the corresponding message region security.

If no values are defined for an MSC link in the IMSMSC field of the IMS control region ACID of the receiving IMS system, no CA Top Secret transaction validation is performed for non-directed routing transactions received on that link.

When MSC transactions are rejected by CA Top Secret because the user does not have the authority to execute them, the user in the originating system receives the message:

DFS2175 TRANSACTION CANCELLED BY MSC LINK EXIT ROUTINE

The complete violation is logged by CA Top Secret and may be found by running a violations report via TSSTRACK or TSSUTIL, on the MVS system where the transaction executed.

The likely causes for the failed transaction are:

- The ACID assigned to the transaction in the target region could not successfully sign on there. This may be due to the ACID being undefined, suspended, or lacking access to some resource, such as the facility.
- The ACID is not permitted to execute the transaction in the target system.

Example: MSC transactions

This example associates region ACID IMS81 with:

- The explicit link name MSC1$2 with propagated user id's from the originating region's terminal session
- The explicit link name MSC4$3 with fixed ACID LINK4$3
- The explicit link name MSC9$8 with the receiving region's facility DEFACID
- The implicit assignment of BADACID unable to sign on in the receiving region to any other link not explicitly defined in previously encountered links

```
TSS ADD(IMS81) IMSMSC('MSC1$2(+USER), MSC4$3(LINK4$3), MSC9$8(+DEFAULT),
ALL(BADACID)')
```

Transactions from APPC

Transactions can enter an IMS region from an APPC conversation. The security processing performed for transaction from APPC is controlled by the values chosen for the APPCSE initialization option.

The APPCSE option can be overridden using the IMS /SECURE command.

If the APPC security options request transaction validation, a SAF call is performed for transactions from APPC that provides CA Top Secret transaction validation in the TIMS resource class.

Transactions from OTMA

Transactions can enter an IMS region from an OTMA client. The security processing performed for transaction from OTMA is controlled by the values chosen for the OTMASE initialization option.

The OTMASE option can be overridden using the IMS /SECURE command..

If the OTMA security options request transaction validation, a SAF call is performed for transactions from OTMA that provides CA Top Secret transaction validation in the TIMS resource class.

Transactions from the OM QUE TRAN Command

Transactions can enter an IMS region from the OM QUE TRAN command. Transactions security for these transactions is the same as for transactions entered from terminals.

Transaction Password Reverification

Transactions that require additional security are defined to require the signon password to be entered with each use. This attempts to prevent certain sensitive transactions from being entered by an unauthorized individual at an unlocked terminal.

Examples: transaction password reverification

This example for LCF transactions requires password reverification to CA Top Secret by coding a V attribute when the transaction is defined to LCF:

```
TSS ADDTO(SUPR01) TRANSACTION(IMSPROD,(QUPD,PROD(V)))
```

This example for OTRAN, password reverification is indicated on the PERMIT by the ACTION (REVERIFY) parameter once the resource is owned:

```
TSS ADDTO(DEPT01) OTRAN(PROD)

TSS PERMIT(SUPR01) OTRAN(PROD)
                   ACTION(REVERIFY)
```

To supply a password with the transaction for either LCF or OTRAN, use the format:

```
trans(password) data
```

For example:

```
PROD(PRNX) SECRET 0447M
```

PROD

An authorized transaction.

PRNX

The user's signon password.

SECRET 0047M

Optional transaction parameters.

IMS Command Security

IMS commands can enter IMS from an:

- IMS terminal
- MCS or EMCS console using the command recognition character defined in the IMS system definition
- Application program using the CMD DLI communications call. These commands are called type 1 AOI commands
- Application program using the ICMD DLI communications call. These commands are called type 2 AOI commands
- APPC conversation
- OTMA client

Implement Command Security

CA Top Secret command security is controlled by the TYPE=RACFCOM parameter of the IMS system definition SECURITY macro, and by the RCF= IMS initialization parameter.

Depending on how the command enters IMS, other IMS initialization options may also affect CA Top Secret command security.

IMS uses a SAF call to invoke CA Top Secret command security. The resource class for these command security calls is formed from the prefix "C" and the value established for the RCLASS parameter (which defaults to "IMS").

CA Top Secret provides a system-supplied resource class CIMS.

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

- If the administrator chooses to use a non-default RCLASS value, the administrator will be responsible for:
- Creating an RDT entry for the resulting command resource class, concatenating "C" with the RCLASS value. Attributes of the new class should be identical with CIMS
- Informing support of the non-standard usage if problems are encountered

The following instructions assume the use of the CIMS facility for command security. The administrator should substitute their non-standard command resource class, if one is in use.

CIMS is a general resource that can be ADDed to establish ownership:

```
TSS ADDTO(acid) CIMS(command)
```

To allow users access to the command, enter:

```
TSS PERMIT(acid) CIMS(command)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) CIMS(command)
```

The first permission allows the user to execute the command only in regions using the IMSPROD facility. The second permission allows the user to execute the command unrestricted by facility.

Note: The resource name in the IMS SAF call that invokes CA Top Secret command security is the first three characters of the command verb, not the entire command verb. When you issue the TSS command to administer command security, the resource name must be limited to these three characters.

This example permits access to the IMS START command:

```
TSS PERMIT(acid) CIMS(STA)
```

Implementation Considerations

When implementing CA Top Secret command security:

- The /SIGN and /RCL commands cannot be protected by CA Top Secret. This is an IMS restriction.
- Administrators should take into consideration common commands that may be required by users to perform their daily tasks, for example:
  - /FORMAT
  - /EXIT
  - /SET
  - /RESET
- Use caution when issuing DEFPROT on the CIMS resource class. This activates security on common commands, unless a global permit has been issued.
- The administrator needs CIMS(OWN) for ADD and REMOVE, and CIMS(XAUTH) for PERMIT and REVOKE.
- All features of owned resources are also supported by CIMS, such as TSS WHOHAS and TSS WHOOWNS. Time and date controls, and scope controls used with the TSS PERMIT command are also supported.
- CA Top Secret does not support the commands:
  ```
  /MODIFY PREPARE RACF
  /MODIFY COMMIT
  ```
  If these commands are issued, IMS issues the message

DFS3400I RETURN CODE 8 FROM RACROUTE MACRO FUNCTION VERIFY.

Commands from Terminals

Command validation is performed for commands entered directly from an IMS terminal.

If the IMS security options request command validation, a SAF call is performed that invokes CA Top Secret command validation.

Type 1 AOI Commands

Application programs can issue IMS commands using the CMD communications call and retrieve the command response using the GCMD communications call. These commands are called type 1 AOI commands.

For releases of IMS prior to IMS r9.1, the only security available for these commands is the TRANCMD security function of SMU.

For IMS r9.1 and above, the AOI1 IMS initialization parameter specifies the level of security for type 1 AOI commands. If the AOI1 security option requests command validation, a SAF call is performed that invokes CA Top Secret command validation for type 1 AOI commands.

Type 2 AOI Commands

Application programs can issue IMS commands using the ICMD communications call and retrieve the command response using the RCMD communications call. These commands are called type 2 AOI commands.

The AOIS IMS initialization parameter specifies the level of security for type 2 AOI commands. If the AOIS security option requests command validation, a SAF call is performed that invokes CA Top Secret command validation for type 2 AOI commands.

Commands from APPC

Commands can enter an IMS region from an APPC conversation. The security processing performed for command from APPC is controlled by the values chosen for the APPCSE initialization option. The APPCSE option can be overridden using the IMS /SECURE command. For information on the /SECURE command, see the IBM *IMS Command Reference*.

If the APPC security options request command validation, a SAF call is performed that invokes CA Top Secret command validation.

Commands from OTMA

Commands can enter an IMS region from an OTMA client. The security processing performed for command from OTMA is controlled by the values chosen for the OTMASE initialization option.

The OTMASE option can be overridden using the IMS /SECURE command.

If the OTMA security options request command validation, a SAF call is performed that invokes CA Top Secret command validation.

AGN Security (IMS r9.1 and Below)

Application group names (AGNs) and AGN security are available in IMS releases up to and including IMS r9.1. AGN security is not available in releases of IMS after r.1. For IMS r.1 and above, dependent region security can be implemented using Resource Access Security (RAS).

An AGN identifies a group of transactions, terminals, and PSBs that an IMS dependent region, such as a BMP or a message-processing region, is permitted to access. AGNs are defined for an IMS system using the Security Maintenance Utility (SMU).

During IMS control region initialization, AGN information is read from the SMU security matrix to determine the AGN names and the resources that are associated with the AGNs.

An IMS dependent region requests the use of an AGN by specifying the AGN name in the startup JCL for the dependent region. During dependent region initialization, the dependent region communicates to the control region the intent to use the AGN. The control region checks to see if the dependent region is permitted access to the requested AGN. If an AGN is not specified in the dependent region JCL or if the ACID is not permitted access to the AGN, IMS terminates the dependent region. If the dependent region is permitted to use the AGN, IMS then limits the dependent region's use of IMS resources to those defined in the AGN that was requested.

AGN Security Administration

CA Top Secret AGN security is controlled by the TYPE=RACFAGN parameter of the IMS system definition SECURITY macro, and by the ISIS= IMS initialization parameter.

CA Top Secret administration for AGNs is done in either the AIMS or the APPLICATION resource class. When CA Top Secret security for AGNs is enabled, IMS issues the AGN security validation with a resource class formed from the prefix "A" and the value established for the RCLASS parameter (which defaults to "IMS"). If the AIMS resource class is defined in the RDT, the AGN security validation is performed in this resource class. If the AIMS resource class is not defined, the AGN resource class is translated to the APPLICATION resource class.

The following instructions assume the use of the APPLICATION resource class for AGN security. The administrator should substitute their non-standard AGN resource class, if one is in use.

APPLICATION is a general resource that can be ADDed to establish ownership:

```
TSS ADDTO(dept) APPLICATION(agn)
```

To allow a dependent region ACID access to the AGN, enter:

```
TSS PERMIT(acid) APPLICATION(agn)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) APPLICATION(agn)
```

The first permission allows the dependent region to specify the AGN only in regions using the IMSPROD facility. The second permission allows the dependent region to specify the AGN unrestricted by facility.

Resource Access Security

Valid on IMS r9.1 and above.

Resource Access Security (RAS) replaces application group names and AGN security. With RAS, CA Top Secret provides direct validation of the terminals, PSBs, and transactions that a dependent region accesses.

CA Top Secret RAS security is controlled by:

- The TYPE=RASRACF parameter of the IMS system definition SECURITY macro
- The ISIS= IMS initialization parameter

Before you enable RAS security:

- Ensure that appropriate security is defined to permit transactions to execute in the message processing regions. If a RAS transaction validation fails in a message processing region, the transaction is not scheduled in the message processing region. This is not a security violation and no violation message is issued. If none of the available message processing regions is permitted to schedule the transaction, the transaction is queued without execution and waits for a message processing region with the appropriate permissions. This can appear to the terminal user as a problem with the transaction or with the IMS system.
- Ensure that the IMS facility is specified in the FACILITY parameter for the batch message processing (BMP) region ACID. Security processing for the BMP region ACID occurs in the IMS control region

RAS Transaction Security

When RAS security is enabled transaction validations occur:

- In message processing regions and Java message regions. Before a transaction is scheduled to execute in the region, IMS performs a security validation to see if the ACID of the region is allowed to execute the transaction. If not, the transaction is not be scheduled in that region.
- In BMPs. If a transaction is read from the IMS message queue, IMS performs a security validation to see if the ACID of the region is allowed to read the transaction.
- In BMPs and Java batch regions. If a transaction is specified in the OUT= parameter in the JCL for the region, IMS performs a security validation to see if the ACID of the region is allowed to generate the transaction.

IMS uses a SAF call to invoke CA Top Secret transaction security. The resource class for these transaction security calls is formed from the prefix "T" and the value established for the RCLASS parameter (which defaults to "IMS"). This is the same resource class that IMS uses for normal transaction validation.

Note: Turn the OPTIONS(27) control option on to translate IMS SAF based TIMS resource class checks to LCF and OTRAN as needed (rather than using TIMS).

CA Top Secret provides a system-supplied resource class TIMS.

Non-default RCLASS Value Use

Rather than use RCLASS to distinguish security permissions for different regions, use separate facilities for distinguished regions and distinguish region-specific permissions by FACILITY.

If the administrator chooses to use a non-default RCLASS value:

- Create an RDT entry for the resulting transaction resource class, concatenating "T" with the RCLASS value. Attributes of the new class should be identical with TIMS
- Inform support of the non-standard usage if problems are encountered

Example: Non-default RCLASS value use

The following instructions assume the use of the TIMS facility for transaction security. Substitute the non-standard transaction resource class, if one is in use.

This example uses the ADDTO command function to add TIMS as a general resource and establish ownership:

```
TSS ADDTO(acid) TIMS(transaction)
```

This example allows the dependent region ACID access to the transaction:

```
TSS PERMIT(acid) TIMS(transaction)
                 FACILITY(IMSPROD)
TSS PERMIT(acid) TIMS(transaction)
```

The first permission allows the dependent region access to the transaction only in regions using the IMSPROD facility. The second permission allows the user to execute the transaction unrestricted by facility.

GIMS Resource Class

The GIMS resource class documented in the IMS product documentation for transaction grouping has no meaning in CA Top Secret. Use profiles for transaction grouping or permit individual transactions in the TIMS resource class.

RAS PSB Security

The following IMS dependent regions can specify a PSB in the dependent region JCL:

- Batch message processing regions (BMPs)
- IMS fast path regions
- Java batch regions

During dependent region initialization, the dependent region communicates to the control region the intent to use the PSB. When RAS security is enabled, the control region checks to see if the dependent region is permitted access to the requested PSB. If the dependent region is not permitted access to the PSB, the dependent region is terminated.

IMS uses a SAF call to invoke CA Top Secret PSB security. The resource class for these security calls is formed from the prefix "I" and the value established for the RCLASS parameter (which defaults to "IMS").

This is the same resource class that IMS uses for program validation for programs on the LOCK and UNLOCK commands. It is not the same as the PSB resource class that CA Top Secret uses for the PSB security provided by the control option IMS(IMSPSBVL).

Use separate facilities to distinguished regions and to distinguish region-specific permissions by FACILITY.

When RAS PSB security is enabled, the security administrator must create an RDT entry for the PSB resource class.

Examples: RAS PSB security

This example assumes the use of the IIMS resource class for PSB security:

```
TSS ADD(RDT) RESCLASS(IIMS)
             RESCODE(xx)
             MAXLEN(8)
```

The following examples assume the use of the IIMS facility for PSB security. Substitute non-standard PSB resource class, if one is in use.

This example uses the ADDTO command function to add an IIMS general resource to establish ownership:

```
TSS ADDTO(acid) IIMS(psbname)
```

This example allows the dependent region ACID access to the PSB:

```
TSS PERMIT(acid) IIMS(psbname)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) IIMS(psbname)
```

The first permission allows the dependent region access to the PSB only in regions using the IMSPROD facility. The second permission allows the user to access the PSB unrestricted by facility.

JIMS Resource Class

The JIMS resource class documented in the IMS product documentation for PSB grouping has no meaning in CA Top Secret. Use profiles for PSB grouping, or permit individual PSBs in the IIMS resource class.

RAS LTERM Security

The following IMS dependent regions can specify an LTERM in the OUT= parameter in the dependent region JCL:

- Batch message processing regions (BMPs)
- Java batch regions

When RAS security is enabled, the IMS control region checks to see if the dependent region is permitted access to the requested LTERM. If the dependent region is not permitted access to the LTERM, the dependent region is terminated.

IMS uses a SAF call to invoke CA Top Secret LTERM security. The resource class for these LTERM security calls is formed from the prefix "L" and the value established for the RCLASS parameter (which defaults to "IMS"). This is the same resource class that IMS uses for validation for LTERMs on the LOCK and UNLOCK commands.

Rather than use RCLASS to distinguish security permissions for different regions, use separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

When RAS LTERM security is enabled, create an RDT entry for the LTERM resource class.

Examples: RAS LTERM security

This example assumes the use of the LIMS resource class for LTERM security:

```
TSS ADD(RDT) RESCLASS(LIMS)
             RESCODE(xx)
             MAXLEN(8)
```

The following examples assume the use of the LIMS facility for LTERM security. Substitute non-standard LTERM resource class, if one is in use.

This example uses the ADDTO command function to add LIMS general resource to establish ownership:

```
TSS ADDTO(acid) LIMS(ltermname)
```

This example allows the dependent region ACID access to the LTERM:

```
TSS PERMIT(acid) LIMS(ltermname)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) LIMS(ltermname)
```

The first permission allows the dependent region access to the LTERM only in regions using the IMSPROD facility. The second permission allows the user to access the LTERM unrestricted by facility.

MIMS Resource Class

The MIMS resource class documented in the IMS product documentation for LTERM grouping has no meaning in CA Top Secret. Use profiles for LTERM grouping, or permit individual LTERMs in the LIMS resource class.

PSB Security

A program specification block (PSB) is an IMS control block that identifies terminals, transactions, databases, and database segments that an application program can access. Because a program that executes online must have the same name as the PSB, transaction security can effectively control the program access to IMS databases.

Controlling the use of the PSB in a batch environment (such as a BMP) is important, because a job can use any PSB with any application program. CA Top Secret provides several control options for the IMS security validation for PSBs. Global control option IMS(IMSPSBVL) enables PSB security in all IMS environments. Control option DL1B(YES) enables PSB security for batch IMS environments.

Note: For PSB security in IMS batch, both DL1B(YES) and IMS(IMSPSBVL) must be set. An IMS(NOIMSPSBVL) specification *disables* PSB protection in all IMS environments. For more information about using options to control IMS security processing, see the *CA Top Secret Control Options Guide*.

Ownership of a PSB immediately protects the resource across all defined IMS regions. The PERMIT function of the TSS command grants access to the PSB. Including the FACILITY keyword as part of the PSB definition limits access to specific regions. Time of day, day of week, access expiration, and ACTION controls are also available.

The ADD function establishes ownership of the PSB:

```
TSS ADDTO(DEPT01) PSB(TSTPAA45)
```

The PERMIT function grants access to the PSB:

```
TSS PERMIT(USER12) PSB(TSTPAA45)
```

Example: Adding PSB Security That Limits the Use of a PSB

This example shows a PERMIT function that includes keywords to limit the use of a PSB:

```
TSS PERMIT(acid) PSB(TSTPAA45) FACILITY(IMSTEST)
                               TIME(13,16)
                               DAYS(WEEKDAYS)
                               FOR(14)
                               ACTION(AUDIT)
```

FACILITY(IMSTEST)

Specifies that the PSB is accessible through a particular facility (IMSTEST). Omission of facility implies access through any defined IMS facility.

TIME(13,16) and DAYS(WEEKDAYS)

Limits PSB access to a time between 1:00 p.m. and 4:00 p.m. during weekdays.

FOR(14)

Provides PSB access authorization for 14 days after the date that the PERMIT command is issued.

Note: To specify a duration that ends at a specific point in time, use the UNTIL keyword. FOR and UNTIL are mutually exclusive.

ACTION(AUDIT)

Audits all accesses to this resource, regardless of the mode or logging options of the user. The ACTION keyword is compatible with all other PERMIT keywords.

Access Levels

When CICS is set up to interface with one or more IMS control regions within the CICS environment, the ACCESS levels INQUIRE, UPDATE, and SET give the user permission to the corresponding CEMT subcommands INQUIRE, UPDATE, and SET, respectively, to manipulate the CICS interface to IMS through the named PSB. (These access levels are not used under an IMS/TM environment.)

DBD Security

A Data Base Descriptor (DBD) is an assembled IMS control block that describes the physical characteristics of a database (including format, contents, and fields of each segment type). The DBD also defines the method of access and the way segment types are physically related.

CA Top Secret provides several options that control the IMS security validation for DBDs. Global control option IMS(IMSDBDVL) enables DBD security in all IMS environments. Control option DL1B(YES) enables DBD security for batch IMS environments.

Note: For DBD security in IMS batch, DL1B(YES) and IMS(IMSDBDVL) must be set. Specifying IMS(NOIMSDBDVL) *disables* DBD protection in all IMS environments. For more information about using options to control IMS security processing, see the *CA Top Secret Control Options Guide*.

DBD security is provided for both logical and physical databases.

Ownership of a DBD immediately protects the resource across all defined IMS regions. The PERMIT function of the TSS command grants access to the DBD. DBDs have access levels associated with them; the access level is derived from the function code on the DL/I call. Specifying the PRIVPGM (Privileged Program) keyword limits DBD access to a particular PSB or group of PSBs. Specifying the FACILITY keyword as part of the DBD definition limits DBD use to specific regions. Time of day, day of week, access expiration, and action controls are also available.

The ADD function establishes ownership of the DBD:

```
TSS ADDTO(DEPT02) DBD(TSTPDA)
```

The PERMIT function allows update access to the DBD:

```
TSS PERMIT(USER13) DBD(TSTPDA)
                   ACCESS(UPDATE)
```

Note: You can also specify ACCESS(INQUIRE) and ACCESS(SET) to use the CEMT transaction in CICS to perform INQUIRE and SET for DBDs.

Example: Apply Limitations to a DBD

This example shows a PERMIT function that includes keywords to apply limitations regarding the DBD:

```
TSS PERMIT(acid) DBD(TSTPDA) FACILITY(IMSPROD) PRIVPGM(TSTPAA45)
                             TIME(09,18)
                             DAYS(MONDAY,WEDNESDAY,FRIDAY)
                             UNTIL(11/24/99) ACTION(FAIL)
```

FACILITY(IMSPROD)

Specifies that the DBD is accessible only through the IMSPROD facility. Omission of FACILITY implies access through any defined IMS facility.

PRIVPGM(TSTPAA45)

Specifies that the DBD is accessible only through program TSTPAA45. In the online environment, an associated transaction would have to specify PSB TSTPAA45 to satisfy the permission. In BMP and DL/I batch environments, MEMBER=TSTPAA45 would have been specified for IMSBATCH or DLIBATCH, respectively.

TIME and DAYS

Limits DBD access to a time between 9:00 a.m. and 6:59:59 p.m. on Monday, Wednesday, and Friday.

UNTIL(11/24/99)

Allows DBD access until November 24, 2099, at which point access is no longer allowed. (You can also use the FOR keyword to specify a duration. FOR and UNTIL are mutually exclusive.)

ACTION(FAIL)

Forces the authorization, if used, to be processed in FAIL mode (regardless of the mode in which CA Top Secret is running globally). In FAIL mode, CA Top Secret requires that all users be defined to CA Top Secret. This mode generates violation messages.

Note: The ACTION keyword is compatible with all other PERMIT keywords.

Resource Security for the IMS /LOCK and /UNLOCK Commands (IMS r9.1 and above)

The IMS LOCK command is used to disable a terminal, stop the scheduling of a specific transaction, stop the scheduling of a specific program, or disable a database, making the IMS resource unavailable for processing. The UNLOCK command can be used to enable the IMS resource again, making it available for processing.

To prevent unauthorized users from disabling IMS resources and affecting IMS processing, security must be provided for the resources specified on the LOCK and UNLOCK commands. In releases of IMS prior to IMS r9.1, IMS provided a mechanism using the IMS Security Maintenance Utility (SMU) to control who was allowed to lock and unlock specific IMS resources.

In IMS r9.1 and above, CA Top Secret can be used to directly protect the resources specified on the LOCK and UNLOCK commands.

CA Top Secret security for resources on the LOCK and UNLOCK commands is controlled by the LOCKSEC= IMS initialization parameter.

Note: This section only pertains to IMS regions running IMS r9.1 and higher. CA Top Secret only provides security for resources specified on the LOCK and UNLOCK command in IMS regions running IMS r9.1 or higher.

Transactions on the IMS /LOCK and /UNLOCK Commands

If LOCK resource security is enabled, when a transaction is specified on a LOCK or UNLOCK command, IMS performs a security validation to see if the user is allowed to LOCK or UNLOCK the transaction.

IMS uses a SAF call to invoke CA Top Secret transaction security. The resource class for these transaction security calls is formed from the prefix "T" and the value established for the RCLASS parameter (which defaults to "IMS").

Notes:

- This is the same resource class that IMS uses for normal transaction validation
- The OPTIONS(27) control option can be turned on in order for the IMS SAF based TIMS resource class checks to be translated to LCF and OTRAN as needed (rather than using TIMS)

CA Top Secret provides a system-supplied resource class TIMS.

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

If the administrator chooses to use a non-default RCLASS value, the administrator will be responsible for the following:

- Creating an RDT entry for the resulting transaction resource class, concatenating "T" with the RCLASS value. Attributes of the new class should be identical with TIMS.
- Informing support of the non-standard usage if problems are encountered.

The following instructions assume the use of the TIMS facility for transaction security. The administrator should substitute their non-standard transaction resource class, if one is in use.

TIMS is a general resource that can be added to establish ownership:

```
TSS ADDTO(acid) TIMS(transaction)
```

To allow the user to LOCK or UNLOCK the transaction, enter:

```
TSS PERMIT(acid) TIMS(transaction)
                 FACILITY(IMSPROD)

TSS PERMIT(acid) TIMS(transaction)
```

The first permission allows the user to LOCK and UNLOCK the transaction only in regions using the IMSPROD facility. The second permission allows the user to LOCK and UNLOCK the transaction unrestricted by facility.

GIMS Resource Class

The GIMS resource class documented in the IMS product documentation for transaction grouping has no meaning in CA Top Secret. You can use profiles for transaction grouping, or permit individual transactions in the TIMS resource class.

Programs on the IMS /LOCK and /UNLOCK Commands

If LOCK resource security is enabled, when a program is specified on a LOCK or UNLOCK command, IMS performs a security validation to see if the user is allowed to LOCK or UNLOCK the program.

IMS uses a SAF call to invoke CA Top Secret program security. The resource class for these program security calls is formed from the prefix "I" and the value established for the RCLASS parameter (which defaults to "IMS").

Note: This is the same resource class that IMS uses for RAS PSB validation.

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

If LOCK resource security is enabled, the security administrator must create an RDT entry for the program resource class.

This example assumes the use of the IIMS resource class for program security.

```
TSS ADD(RDT) RESCLASS(IIMS)
             RESCODE(xx)
             MAXLEN(8)
```

The following instructions assume the use of the IIMS facility for program security. The administrator should substitute their non-standard program resource class, if one is in use.

IIMS is a general resource that can be ADDed to establish ownership:

```
TSS ADDTO(acid) IIMS(program)
```

To allow the user to LOCK or UNLOCK the program, enter:

```
TSS PERMIT(acid) IIMS(program)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) IIMS(program)
```

The first permission allows the user to LOCK and UNLOCK the program only in regions using the IMSPROD facility. The second permission allows the user to LOCK and UNLOCK the program unrestricted by facility.

JIMS Resource Class

The JIMS resource class documented in the IMS product documentation for program grouping has no meaning in CA Top Secret. You can use profiles for program grouping, or permit individual programs in the IIMS resource class.

LTERMs on the IMS /LOCK and /UNLOCK Commands

If LOCK resource security is enabled, when an LTERM is specified on a LOCK or UNLOCK command, IMS performs a security validation to see if the user is allowed to LOCK or UNLOCK the terminal.

IMS uses a SAF call to invoke CA Top Secret LTERM security. The resource class for these LTERM security calls is formed from the prefix "L" and the value established for the RCLASS parameter (which defaults to "IMS").

Note: This is the same resource class that IMS uses for RAS LTERM validation.

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

If LOCK resource security is enabled, the security administrator must create an RDT entry for the LTERM resource class.

This example assumes the use of the LIMS resource class for LTERM security:

```
TSS ADD(RDT) RESCLASS(LIMS)
             RESCODE(xx)
             MAXLEN(8)
```

The following instructions assume the use of the LIMS facility for LTERM security. The administrator should substitute their non-standard LTERM resource class, if one is in use.

LIMS is a general resource that can be ADDed to establish ownership:

```
TSS ADDTO(acid) LIMS(lterm)
```

To allow the user to LOCK or UNLOCK the terminal, enter:

```
TSS PERMIT(acid) LIMS(lterm)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) LIMS(lterm)
```

The first permission allows the user to LOCK and UNLOCK the terminal only in regions using the IMSPROD facility. The second permission allows the user to LOCK and UNLOCK the terminal unrestricted by facility.

MIMS Resource Class

The MIMS resource class documented in the IMS product documentation for LTERM grouping has no meaning in CA Top Secret. You can use profiles for LTERM grouping, or permit individual LTERMs in the LIMS resource class.

Databases on the IMS /LOCK and /UNLOCK Commands

If LOCK resource security is enabled, when a database is specified on a LOCK or UNLOCK command, IMS performs a security validation to see if the user is allowed to LOCK or UNLOCK the database.

IMS uses a SAF call to invoke CA Top Secret database security. The resource class for these database security calls is formed from the prefix "P" and the value established for the RCLASS parameter (which defaults to "IMS").

Note: This is the same resource class that IMS uses for database authorization for the IMS application AUTH call.

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to make use of separate facilities for distinguished regions and to distinguish region-specific permissions by FACILITY.

If LOCK resource security is enabled, the security administrator must create an RDT entry for the database resource class. The following example assumes the use of the PIMS resource class for database security.

```
TSS ADD(RDT) RESCLASS(PIMS)
             RESCODE(xx)
             MAXLEN(8)
```

The following instructions assume the use of the PIMS facility for database security. The administrator should substitute their non-standard database resource class, if one is in use.

PIMS is a general resource that can be ADDed to establish ownership:

```
TSS ADDTO(acid) PIMS(database)
```

To allow the user to LOCK or UNLOCK the database, enter:

```
TSS PERMIT(acid) PIMS(database)
                 FACILITY(IMSPROD)
```

```
TSS PERMIT(acid) PIMS(database)
```

The first permission allows the user to LOCK and UNLOCK the database only in regions using the IMSPROD facility. The second permission allows the user to LOCK and UNLOCK the database unrestricted by facility.

QIMS Resource Class

The QIMS resource class documented in the IMS product documentation for database grouping has no meaning in CA Top Secret. You can use profiles for database grouping, or permit individual databases in the PIMS resource class.

OTMA Resume TPIPE Request Security

If you are using OTMA in an IMS environment with IMS r10 and above, you can protect messages on OTMA asynchronous hold queues from unauthorized use of the RESUME TPIPE request.

If security is enabled for the TPIPE hold queues, the userid issuing a RESUME TPIPE request must be authorized to access the TPIPE name in the RESUME TPIPE request before TPIPE messages are sent to the OTMA client.

IMS uses a SAF call to invoke CA Top Secret TPIPE security. The resource class for these TPIPE security calls is formed from the prefix R and the value established for the RCLASS parameter (defaults to IMS).

Rather than use RCLASS to distinguish security permissions for different regions, CA Top Secret encourages the administrator to use separate facilities for distinct regions and to distinguish region-specific permissions by FACILITY.

To enable TPIPE security, the security administrator must create an RDT entry for the TPIPE resource class.

Examples: TPIPE security

This example assumes the use of the RIMS resource class for TPIPE security:

```
TSS ADD(RDT) RESCLASS(RLIMS)
             RESCODE(xx)
             MAXLEN(8)
```

The following instructions assume the use of the RIMS facility for TPIPE security. If in use, substitute any non standard TPIPE resource class.

This example establishes ownership of the RIMS is a general resource:

```
TSS ADDTO(acid) RIMS(tpipe)
```

This example allows the user to issue the RESUME TPIPE request for the TPIPE only in regions using the IMSPROD facility:

```
TSS PERMIT(acid) RIMS(tpipe)
                 FACILITY(IMSPROD)
```

This example allows the user to issue the RESUME TPIPE request for the TPIPE unrestricted by facility.

```
TSS PERMIT(acid) RIMS(tpipe)
```

Terminal Security

Terminals are owned resources; therefore, auditing and distributed administration can easily be performed. Ownership of a terminal protects the resource across all defined facilities; however, access can be limited to only specific facilities through the FACILITY keyword as part of the terminal definition. Time of day, day of week, access expiration, and action controls are also available.

The terminal name defined depends on your environment:

- IMS PTERM definition for all VTAM environments
- IMS LTERM definition for all BTAM environments and LU6 terminals

Notes:

- Terminal protection applies across all facilities with like terminal names. IMS PTERM definition will also limit VTAM terminals for facilities such as TSO. Be sure to check the implication of terminal security as related to TSO, CICS, or any other online facilities under CA Top Secret control..
- The FACILITY keyword specifies that the terminal can only be accessed through a particular facility, such as IMSTEST. Omission of FACILITY implies access through any defined IMS facility.
- Terminal and source restriction is in effect in the originating IMS region for MSC transactions. Terminal protection is not in effect in the target IMS region.
- ATS users are not automatically PERMITted access to owned terminal resources on the basis of their name alone.

Terminal Session Lock

CA Top Secret allows individual users to lock their terminal session. While locked, a session will allow only the IMS commands shown below which terminate the current user's session.

```
/SIGN
```

```
/RCL
```

The only transaction format a locked session will recognize is one with a reverification password-whether or not the transaction normally is designed to use the password.

To directly unlock the session, enter:

```
TSS(password) UNLOCK
```

This example implicitly unlocks the session while explicitly invoking the transaction "trans";

```
trans(password) data
```

Considerations for the IMS AUTH Call

The IMS AUTH call is used to inquire against the standard security interface (SSI), so that a program can check IMS internal and external security for the signed-on user before making a request for resources.

The AUTH call makes security requests on behalf of the signed-on user. If the resource is defined in the SECURITY gen, no external security check is performed. The request will be allowed with an IOPCB status of spaces or denied with a status of A4, depending entirely on the compiled rules genned into the region's MATRIX data sets.

This example defines and permits PIMS:

```
TSS ADDTO(RDT) RESCLASS(PIMS)
               RESCODE(xx)
               MAXLEN(8)
```

```
TSS ADDTO(dept) PIMS(dbdname)
```

```
TSS PERMIT(acid) PIMS(dbdname)
```

If the AUTH standard or generated RESCLASS is used in CA Top Secret, it is used only for these application AUTH calls. These resource classes are not used for CA Top Secret IMS resource security.

Use the CA Top Secret Application Interface for applications-based authorization processing. This interface uses the same resource classes and permissions used to protect actual IMS access.

If you use the IMS AUTH call for application-based resource security, translate the IMS generated resource classes into traditional TSS defined resource classes. To implement resource translation, use the RXLTADD operand to add translation entries to the appropriate IMS facility definition.

This example translates IMS generated TIMS transaction security requests to the traditional TSS OTRAN resource class for the IMSPROD facility:

```
TSS MODIFY FACILITY(IMSPROD=RXLTADD(TIMS:OTRAN))
```

After this command is executed, resource security calls with the TIMS resource class are checked against OTRAN user permissions.

Resource Translation

CA Top Secret control option OPTION(27) can be used for resource translation. If OPTIONS(27) is specified, CA Top Secret converts the IBM transactions resource class TIMS to the traditional LCF or OTRAN transaction resource class. Because it specifically translates the IMS standard TIMS resource class, the standard IMS SECURITY macro RCLASS=IMS is required. This option does not translate any of the other AUTH resource types.

If IMS AUTH calls are used for application based resource security and no resource class translation is performed, the administrator is responsible for security rules in the IMS generated resource class for the AUTH call resources, and in the traditional CA Top Secret resource classes for CA Top Secret IMS security.

Undefined Resources

If the resource is not defined in the SECURITY gen, IMS will make requests through the standard security interface based on the CLASSNAME parameter in the AUTH call I/O area as follows:

I/O Area CLASSNAME	Standard RESCLASS	Generated RESCLASS
TRAN	TIMS	Txxx
DATABASE	PIMS	Pxxx
SEGMENT	SIMS	Sxxx
FIELD	FIMS	Fxxx
OTHER	OIMS	Oxxx

I/O Area CLASSNAME

Defined by IBM documentation for the AUTH call.

Standard RESCLASS

Based on the standard default value RCLASS=IMS in the Stage 1 SECURITY macro.

Generated RESCLASS

If you specify a non-default value, that is RCLASS=xxx, the AUTH call will generate the RESCLASS for its security queries.

CA Top Secret does not by default supply support for either the Standard or Generated RESCLASS used by the AUTH call. If you choose to define these resource classes directly to CA Top Secret, you must define them to the RDT. In addition, you must provide separate security permissions for these inquiry resource classes, which can duplicate security already present for LCF, OTRAN, PSB or DBD security.

Administration Using the TSS Transaction

You can use the TSS transaction through IMS for all administration and auditing functions. All changes made via the TSS command in any facility are effective immediately across all address spaces (CCI will transfer these to related CPUs). If the change is made to a user already signed on, the user or the security administrator may issue the TSS REFRESH command. This will refresh the security environment so that the user does not have to log off and back on to access the new environment. It is your responsibility to install this transaction as part of your generation.

Application Interface

CA Top Secret provides a security call and verification interface for use by IMS applications. The interface can be used for:

- Extraction and update of installation-defined data and retrieval of signon name information
- Security checks against IMS or user-defined resources
- Automatic logging of violations
- Logging of user-defined information

Checks against unauthorized resources do not cause failure of the task. Handling of the violation belongs to the application program.

Invoke the Application Interface

To use the CA Top Secret IMS application interface, security must be properly installed. In an IMS TM environment, this means that message region security must be active. Message region security is controlled by the IMS parameter ISIS=.

In a DLIBATCH environment, security is controlled by the CA Top Secret DL1B control option; this option must be set to YES for the application interface to be properly secured.

If security is not properly installed, U0476 abends will occur when IMS attempts to process the application interface calls.

To invoke the CA Top Secret IMS application interface, the application program must issue a DL/I call specifying a special function code of TSS, and passing a request record that describes the processing to be performed by the application interface.

The request record is essentially a parameter list containing fields supplied by the application program and return areas which will be filled in by CA Top Secret upon return from the request. This record is supplied as the second parameter on the DL/I call.

Note: All fields in the request record should be initialized before the call is made.

In IMS, a partial list of resource classes that can be checked using the request record includes:

- ABSTRACT
- APPL
- DBD
- FIELD
- PSB
- TERMINAL
- UR1
- UR2

COBOL Example

```
1      78
000010 IDENTIFICATION DIVISION .
           .
001000 DATA DIVISION .
           .
001010*
001020*
001030* ENSURE TSSCPLC IS IN THE COBOL COPY LIBRARY OR COPY YOUR OWN
001040* VERSION INLINE INTO PROGRAM, USE OF COPY LIBRARY IS
001050* RECOMMENDED SO THAT CHANGES WILL ONLY REQUIRE RECOMPILE
001060* AND NOT RE-CODING OF SOURCE
001070* COPYBOOK CAN BE FOUND IN THE OPTIONAL MATERIALS DATASET
001080 01    TSS                 PIC X(4) VALUE 'TSS '.
001090 01 TSSCPLC COPY TSSCPLC REPLACING ==:RTLN:==BY==256==.
           .
002000 PROCEDURE DIVISION .
           .
           .
010010*
010020*
010030* CALL CA Top Secret APPLICATION INTERFACE TO EXTRACT
010040* INSTALLATION DATA FROM USERS SECURITY RECORD
010050*
010060*
010065 MOVE 'TCPLV5L1' TO TSSHEAD
010070 MOVE 'DUFXTR' TO TSSCLASS.
010080 MOVE 'TSSAI'  TO TSSAI.
010090 CALL 'CBLTDLI' USING TSS , TSSCPL.
010100 IF  NOT TSSROK GO TO 100-CHECK-TOP-RETURN.
010110 MOVE TSSRTN  TO USER-AREA-FOR-INST-DATA.
           .
020010*
20020*
020030* CANCEL CA Top Secret APPLICATION INTERFACE. DO THIS WHEN THE
020040* CA Top Secret INTERFACE IS NO LONGER REQUIRED BY PROGRAM AND
020050* THUS FREE UP THE STORAGE IT OCCUPIES IF NOT IN COMMON STORE
020060*
020070*
020090 MOVE 'TSSAI'  TO TSSAI.
020100 CANCEL TSSAI.
```

Assembler Example

```
COPY  TSSCPLA                 MAKES MACRO AVAILABLE
TSS        DC    CL4'TSS '               DLI CALL REQUEST
TSSCPL     #TSSCPLA TRLN=2048            APPLICATION INTERFACE REQUEST
           .
           .
           .
          CALL  ASMTDLI,(TSS,TSSCPL),VL CALL APPLICATION INTERFACE
```

PL/I Example

```
PLIEX1: PROC OPTIONS(MAIN);
           .
           .
           .
%INCLUDE(TSSCPLP);                         /* APPLIC INTRF PLIST  */
TSSRTN□CTL=2048;
ALLOCATE TSSCPL;
                        /* APPLIC INTRF PLIST  */
DCL 1 TSS CHARACTER(04) INIT('TSS ');      /* INDICATE TSS REQ    */
DCL 1 PCOUNT STATIC FIXED BIN(31) INIT(2); /* CONT OF PARMS AFTER */
           .
           .
           .
          CALL PLITDLI (PCOUNT,TSS,TSSCPLP)
```

Chapter 6: Implementing Security for TSO

This section contains the following topics:

TSO and CA Roscoe Considerations

Considerations for TSO and CA Roscoe are similar to BATCH considerations because they deal with the same resources. Plan the implementation strategy to incorporate the BATCH implementation with the TSO or CA Roscoe implementation.

Important! For TSO users, an ACID can have a maximum of seven characters.

A special control available with both TSO and CA Roscoe is the ability to control access to TSO and CA Roscoe commands, SPF/ISPF panels, and CA Roscoe monitors through the Limited Command Facility (LCF). This lets you specify inclusive and exclusive lists of commands, panels, and/or monitors available to a user in each facility.

The CA Top Secret UADS conversion program TSSCVUAD runs as a one-time execution creating TSS commands that add all the users on UADS to the security file. You can then add all subsequent new TSO users directly to the file.

A special consideration with CA Roscoe is that your plans must include maintenance of the CA Roscoe User Profile data set. While CA Top Secret is replacing the password controls available with this facility, this table is still required by CA Roscoe for certain management routines.

Installation of TSO

For typical installations, interfacing CA Top Secret to TSO/E does not demand special installation steps. If you are running under a non-TSO/E environment, you must specify the TSO operand for the PRODUCTS control option.

Access Operation

Facilities defined in the CA Top Secret Facility Matrix Table default to FAIL mode. To avoid locking users out, an CA Top Secret administrator can edit the Parameter File and change the MODE parameter to DORMANT prior to starting or restarting CA Top Secret. The syntax for making this change is:

```
FACILITY(TSO=MODE=DORM)
```

Predicted System Action

The following table summarizes access operation by mode for use in predicting system action.

FUNCTION	DORM MODE	WARN MODE	IMPL MODE	FAIL MODE
Signon Processing Required		X	X	X
Full Password Controls	1	2	2	X
Transaction Security Active		3	X	X
Terminal Protection Active		3	X	X
TSS Command Available	X	X	X	X
Application Interface Available	X	X	X	X
Logging Available		X	X	X
Auditing Available		X	X	X

X

All Users in All Conditions

1

If Explicit Signon is Performed

2

For Defined Users Only

3

Messages and Logging Only

Note: Security Control Administrators, that is, MSCAs, SCAs, LSCAs, ZCAs, VCAs, DCAs, must use their CA Top Secret password in any mode.

Violation Thresholds

Terminal locking due to excessive violations is enforced through CA Top Secret. If the VTHRESH control option is in effect and the action is either CAN (for cancel) or SUS (for suspend), then CA Top Secret will lock the terminal when the threshold is exceeded. The user's only course of action is to sign off. If the action is VTHRESH(SUS), then the user will be automatically suspended and will not be able to sign on until unsuspended by a security administrator.

Auditing Options

Auditing of all defined users and ownable resources is accomplished using CA Top Secret logging and auditing mechanisms.

Examples: auditing options

This example adds the AUDIT attribute to the ACID for defined users:

```
TSS ADDTO(USER05) AUDIT
```

This example adds the resource name or prefix to the AUDIT record for ownable resources (APPLICATION, PSB, DBD, and Terminal):

```
TSS ADDTO(AUDIT) APPLICATION(TSTPAY)
```

TSO Versus TSO/E

CA Top Secret defaults to TSO/E. If your installation runs under TSO, you must use the PRODUCTS control option operand TSO to identify TSO to CA Top Secret, for example:

```
PRODUCTS(TSO,prod1,prod2,...,prodn)
```

prod1,prod2,...,prodn

The names of other installed products.

When placed in the Parameter File, the TSO version specified in the PRODUCTS control option overrides TSO/E.

TSO Security

Under TSO and TSO/E, CA Top Secret governs how ACIDs and ACID passwords can be defined. In addition, CA Top Secret TSO provides security in the following areas:

- ISPF menu security
- Terminal security
- Job submission control
- TSO resource security

ACIDs to TSO/E Definition

With TSO/E version 2, CA Top Secret can serve as a repository for TSO user logon information, thus eliminating the need for the TSO User Attribute Data Set (UADS). For an ACID to be allowed access to these releases of TSO/E:

- A CA Top Secret ACID must be created for the user.
- Either the ACID must have a TSO UADS userid defined via the TSO ACCOUNT command or at least *one* of the user TSO data fields-TSOLPROC, TSOLACCT, and so on- must have been added to the ACID
- The ACID must be authorized to use TSO via the FACILITY keyword.

Password Controls

All users, except those defined with an CA Top Secret password of NOPW (no password), must enter a correct password at logon before they can access TSO.

Password Verification

TSO logon password verification is always active, even during the CA Top Secret implementation stages. However, if the CA Top Secret address space is inactive, password verification is controlled by the value assigned to the DOWN control option.

CA Top Secret performs password verification for:

- Any ACID which has TSO data fields as part of his Security Record- regardless of the mode.
- Defined users in DORMANT mode if the DORMPW facility option is set.
- Defined users in WARN mode if the WARNPW facility option is set. (WARNPW is the default option for TSO.)
- Defined users in IMPL modes.
- All users in FAIL mode.
- All Security Control Administrators in ALL modes.

MVS performs normal TSO UADS password verification for:

- Undefined users in IMPL and WARN modes.
- All users (except Security Control Administrators and ACIDs which have TSO data fields as part of their Security Records) in DORMANT mode.
- All users in BYPASS mode.
- All users with the TSOMPW attribute.

When a user *reconnects* to TSO after losing an ACF/VTAM terminal, CA Top Secret validates the userid, the password, and the terminal being used.

Note: For non-TSO/E environments, the password will not be honored with the LOGON command if enforced password prompting is in effect.

Split-word passwords should generally not be used if your site uses TSO LOGON Reconnect. If you require split-word processing, you must disable LOGON Reconnect by setting the TSO LOGON Reconnect interval to 0.

Password Signon Rules: UADS Versus CA Top Secret

Provided that the CA Top Secret user attribute TSOMPW has *not* been set to support UADS multiple passwords, the following rules outline which password should be entered at logon:

- Security Control Administrators, that is MSCAs, SCAs, LSCAs ZCAs, VCAs, and DCAs, use their CA Top Secret password in any mode.
- User ACIDs defined to CA Top Secret use:
  - CA Top Secret passwords in IMPL and FAIL modes.
  - CA Top Secret passwords in all modes if the ACID has TSO data fields as part of his Security Record.
  - CA Top Secret passwords in WARN mode if the FACILITY WARNPW option is set or in DORMANT mode if the FACILITY DORMPW option is set.
  - UADS passwords in DORMANT mode.
- Users not defined to CA Top Secret:
  - Use their UADS password in DORMANT, WARN, or IMPL modes
  - Are not allowed system access in FAIL mode

Password Prompting

The PROMPT and NOPROMPT settings discussed in this section apply to TSO/E users who use the TSO/E Logon Panels and supply their password in the indicated field.

The desired method of password entry must be defined in the CA Top Secret Parameter File by the PROMPT operand of the FACILITY control option. There are two settings:

- FACILITY(TSO=PROMPT) Indicates that TSO users must wait for CA Top Secret to prompt for a password. Passwords are entered in a non-display field, which provides greater security.
- FACILITY(TSO=NOPROMPT)

The default is NOPROMPT, which allows TSO users to enter their passwords with the LOGON command.

Penalty for Supplying Password

If the FACILITY(TSO=PROMPT) feature is in effect and a TSO user enters his password with the LOGON command, CA Top Secret locks the user's terminal for approximately ten seconds, issues a message stating that passwords should not be entered with the userid, then prompts the user for a password.

The delay in LOGON execution time, followed by the required reentry of the password, are designed to discourage users from entering their passwords with the LOGON command.

Logon Steps

When FACILITY(TSO=PROMPT) is active, a TSO user must log on in the following manner:

1. Enter:

   ```
   LOGON acid
   ```

   acid

   Your CA Top Secret ACID.

   CA Top Secret displays the password prompt.

2. Type your password in the non-display field.

FACILITY(TSO=NOPROMPT)

If the FACILITY(TSO=NOPROMPT) feature is in effect:

- Users can enter their passwords with the LOGON command
- Passwords are displayed, making them easily viewable.

The format for LOGON is:

```
LOGON acid/password
```

Multiple Password Support

By default, CA Top Secret does not support UADS structures that allow users to have multiple passwords. Using the CA Top Secret attribute, TSOMPW, administrators can make multiple password support available on a user-by-user basis.

The TSOMPW attribute can be added to a user ACID or a profile ACID.

This example indicates that CA Top Secret will support multiple passwords for USER01:

```
TSS ADDTO(USER01) TSOMPW
```

Note: This attribute is not honored for ACIDs whose Security Records contain TSO logon data fields. It is also not valid for Security Control Administrators, that is MSCAs, SCAs, LSCAs, ZCAs, VCAs, and DCAs.

If the TSOMPW attribute has been added to a user or a profile, the user must enter his CA Top Secret password first. After entering his CA Top Secret password, the user is asked to enter their UADS passwords.

CA suggests that the UADS password be changed to represent a function. That way, a TSO user logs on to a particular function indicated by the UADS password.

Change Your Password

Provided that the “No Password Change” (NOPWCHG) attribute has not been associated with a TSO user and the “New Password” (NEWPW) control option is not set to NU, users can change their passwords during any LOGON session.

Note: If your password expires, CA Top Secret automatically prompt (or have TSO/E prompt) users for a new password.

Password Expiration for TSO/E

Passwords are valid for the interval of time set by the CA Top Secret PASSWORD attribute. When a password is about to expire, CA Top Secret warns the user:

TSS7003W PASSWORD WILL EXPIRE ON mm/dd/yy.

If desired, the user may enter his new password (or RANDOM) at the time this message is displayed or wait until the expiration date; either way, CA Top Secret will force the user to supply a new password.

If you are running TSO/E, and random password generation is not being used, the following events occur on the password expiration date.

- The user supplies his valid signon ID and old password.
- CA Top Secret displays this message:

  TSS7110E PASSWORD HAS EXPIRED. NEW PASSWORD MISSING.

- After a two second delay, CA Top Secret will return control back to TSO/E which positions the cursor at the NEW PASSWORD field on the logon menu.
- If random password generation is not mandatory, but it is supported, the user has the default options of:
  - Entering a new password, or
  - Entering RANDOM, causing CA Top Secret to randomly generate one.

To make password reverification mandatory, security administrators can use the NPWR suboption of the FACILITY control option. With the NPWR suboption in place, users who specify their own new passwords will be forced to reenter the new password. If the user tries two times (or the number of times indicated by the NPWRTHRESH control option) and cannot reenter the matching password, CA Top Secret issues a TSS7111E message, telling the user that his new password change is invalid.

Password Expiration for Non-TSO/E

If you are not running TSO/E, and random password generation is not being used, the following events occur on the password expiration date:

- The user supplies his valid signon ID and old password.
- CA Top Secret displays these messages:

  TSS7110E PASSWORD HAS EXPIRED. NEW PASSWORD MISSING.

  TSS7012W * WARNING *** NEVER USE PERSONAL INFORMATION AS YOUR PASSWORD.**

  TSS7013W INCLUDING NAMES,LOCATIONS,LICENSE/PHONE NUMBERS, SSN/SIN.

  TSS7011A PLEASE Enter YOUR *NEW* PASSWORD OR “RANDOM”.

- The user must supply a new password.
- If random password generation is not mandatory, but it is supported, the user has the default options of:
  - Entering a new password or
  - Entering RANDOM, causing CA Top Secret to randomly generate one.
- The user must specify a new password using one of these syntaxes:

  newpassword/newpassword

  Takes advantage of the CA Top Secret voluntary reverification option (newpassword must match newpassword)

  newpassword

  Circumvents the voluntary reverification option.

To make password reverification mandatory, security administrators can use the NPWR suboption of the FACILITY control option. With the NPWR suboption in place, users who specify their own new passwords will be forced to reenter the new password. If the user tries two times (or the number of times indicated by the NPWRTHRESH control option) and cannot reenter the matching password, CA Top Secret issues a TSS7111E message, telling the user that his new password change is invalid.

Random Password Generation

TSO supports random generation of passwords. Automatic generation of random passwords is provided only if NEWPW(RN) is set. A random password is generated at the first logon with an expired password.

ISPF Menu Option Protection

CA Top Secret allows an administrator to restrict the use of SPF Panel Options. This protection is only available for the IBM-supplied SPF functions.

CA Top Secret considers each SPF panel option a TSO command.

For example, CA Top Secret is able to associate VTOC display (SPF option 3.7) with the command SPF37. Therefore, VTOC can be protected by restricting the use of the TSO command, SPF37, by specifying:

```
TSS ADDTO(profile) XCOMMAND(TSO,SPF37)
```

The format for specifying SPF and ISPF options is:

SPF*n*

n

The SPF menu number.

CA Top Secret protects the SPF panels listed in the following table, provided these panels invoke the same corresponding options. The panels and options in the table represent the standard IBM SPF panels and options.

SPF Menu Number	Option
SPF0	TERMINAL OPTIONS
SPF1	BROWSE
SPF2	EDIT
SPF3	LIBRARY or DATA SET
SPF33	LIBRARY or DATA SET
SPF34	CATALOG or SPF3.4 UTILITY
SPF35	RESET
SPF36	HARDCOPY
SPF37	VTOC
SPF38	OUTLIST
SPF4	FOREGROUND
SPF5	BACKGROUND
SPF6	TSO
SPF7	SUPPORT
SPF71	TEST PANEL

SPF Menu Number	Option
SPF72	TEST FUNCTIONS
SPF73	TEST VARIABLES
SPF74	CONVERT MENUS
SPF75	CONVERT MESSAGES
SPF76	TEST MENUS

Terminal Locking

The automatic terminal locking feature is available for TSO. 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.

CA Top Secret sees the time difference between command executions only. It does not see keystrokes or recognize when a user presses the Enter key. This is important for users who spend long periods of time in EDIT or SPF/EDIT. Use the timeout feature of your session manager if the LOCKTIME feature in the TSO does not meet you needs.

If a user has been in EDIT or SPF/EDIT, his terminal may lock when he ends the edit session. If this occurs, the user must enter the TSS UNLOCK command by switching to menu 6, or under TSO/E, on the menu command line.

Note: When TOP SECRET locks the terminal, TSO responds with a S806 abend.

A locked terminal can be logged off. Also, terminal locking does not interfere with the MVS timeout feature (S522 abend).

BATCH TMP

BATCH TMP (Terminal Monitor Program) can be used to execute groups of TSS commands through TSO CLISTs, data files, or direct input to the SYSTSIN DD statement.

The following JCL example, when submitted through BATCH TMP, defines a group of users to CA Top Secret.

```
//TSSDEFIN JOB  USER=sca,MSGCLASS=A
//         EXEC PGM=IKJEFT01
//SYSTSPRT DD   SYSOUT=*,
//         DCB=(LRECL=133,BLKSIZE=133,RECFM=FA)
//* Define the following ACIDS to CA Top Secret */
//SYSTSIN DD *
TSS CREATE(USER01) NAME(HALSE) FAC(TSO) DEPT(TWRT) +
PASSWORD(XXXX,22,EXPIRED) TYPE (USER)
TSS CREATE(USER02) NAME(PARRI) FAC(TSO) DEPT(TWRT) +
PASSWORD(XXXX,22,EXPIRED) TYPE(USER)
TSS CREATE(USER03) NAME(GEACH) FAC(TSO) DEPT(TWRT) +
PASSWORD(XXXX,22,EXPIRED) TYPE(USER)
TSS CREATE(USER04) NAME(GAMBI) FAC(TSO) DEPT(TWRT) +
PASSWORD(XXXX,22,EXPIRED) TYPE(USER)
TSS CREATE(USER05) NAME(GORDO) FAC(TSO) DEPT(TWRT) +
PASSWORD(XXXX,22,EXPIRED) TYPE(USER)
TSS CREATE(USER06) NAME(WANGE) FAC(TSO) DEPT(TWRT) +
PASSWORD(XXXX,22,EXPIRED) TYPE(USER)
/*
```

BATCH Job Submission

BATCH is a separate facility that users must be authorized to use through TSS CREATE or ADD. Defined users are allowed to submit only under their ACID. To authorize an ACID to submit jobs under other ACIDs, use the following CA Top Secret command:

```
TSS PERMIT(acid1) ACID(acid2).
```

acid1

The ACID receiving the authorization and acid2 is the ACID of another ACID. Or can add NOSUBCHK to allow acid to submit jobs under any acid.

Batch jobs submitted from TSO are associated with an ACID in one of two ways:

- A USER= parameter on the job card
- SUBACID control option for jobs submitted online via TSO.

CA Top Secret will try to find a valid ACID for the job by searching in the order described above. If it cannot locate a valid ACID, the job initiation will fail.

TSO Resource Use Restriction

By default, users defined to the TSO facility will have access to all TSO commands and programs until restrictions are placed on an individual or group of users.

CA Top Secret provides three methods for protecting TSO commands or command subsets:

- Use FACILITY(TSO=XDEF) to protect all TSO commands,
- Use the Limited Command Facility (LCF) to restrict use of special TSO commands, or
- Protect TSO commands as PROGRAMS.

Certain TSO commands remain unconditionally accessible to all ACIDs. Restrictions cannot be placed on the TSO commands listed below:

- END
- HELP
- LOGOFF
- LOGON
- PROFILE
- TERMINAL
- WHEN

The TSO CALL command cannot be restricted. To protect a program that is CALLed, place the name of the program in the LCF list.

FACILITY(TSO=XDEF)

All TSO commands can be restricted through the FACILITY control option operand XDEF. If your installation specifies FACILITY(TSO=XDEF), all TSO commands must be defined to a user or profile through an LCF CMD list before they can be used.

By default, TSO commands are authorized to all users via the FACILITY control option operand, NOXDEF. If your installation prefers to restrict TSO commands, you must change the FACILITY(TSO=NOXDEF) parameter to FACILITY(TSO=XDEF).

Limited Command Facility (LCF)

The Limited Command Facility (LCF) consists of the CA Top Secret keywords:

COMMAND

An inclusive LCF list.

XCOMMAND

An exclusive LCF list.

Due to the CA Top Secret algorithm for settling command authorization discrepancies, you should select either COMMAND or XCOMMAND to restrict TSO commands. Otherwise, if an ACID's Security Record lists both a COMMAND and an XCOMMAND restriction for the same TSO command, COMMAND overrides XCOMMAND. In that situation, CA Top Secret ignores the XCOMMAND restriction and allows access.

LCF is also the vehicle for replacing the command limiting functions of the PCF program product. Use either inclusive (the COMMAND keyword) or exclusive (XCOMMAND keyword) command lists to limit usage.

Protect TSO Commands as PROGRAMS

TSO commands execute specific programs. To eliminate LCF lists (which protect the TSO commands), you can protect the programs executed by the commands. An advantage of using program protection for IBM programs, is the insurance that an ACID does not gain access to a TSO command due to an LCF list restriction discrepancy.

Ownership provides protection. Adding a program to an ACID provides ownership. Programs must be owned on a divisional or departmental level. As a result, users must be authorized before they can access the program.

```
To protect a program, enter:
```

```
TSS tssfunc(acid) PROGRAM(ppp,ppp,...,ppp)
```

tssfunc

A TSS function, that is, ADD, CREATE.

acid

A particular ACID.

ppp

Program name or prefix.

Example: protecting a program

In this example all programs prefixed by IEH are protected:

```
TSS ADDTO(CORPDV) PROGRAM(IEH)
```

Access to Owned Data Sets

TSO users do not automatically have access to data sets starting with their TSO userid (ACID). Two common methods used to allow access are:

- Establish ownership by ADDing the DSNAME prefix to the user's ACID:

  ```
  TSS ADDTO(acid) DSNAME(acid.)
                  ACCESS(ALL)
  ```

- Use:

  ```
  TSS PERMIT(ALL) DSNAME(%.)
                  ACCESS(ALL)
                  FACILITY(TSO)
  ```

The prefix “%.” is a masking character allowing the use of data sets beginning with a user's ACID. Note that, if masking characters are used, the prefix, (%.), must be owned by the MSCA acid before any PERMITs can be issued.

UADS Replacement

CA Top Secret provides TSO default data fields and TSO-related resources so you can replace UADS.

TSSCVUAD Conversion Program

CA Top Secret UADS conversion program (TSSCVUAD) runs as a one-time execution, creating TSS commands which can be used as input to the batch TMP. Once the commands have been created, users defined in the UADS data set, as well as their associated UADS data fields, are defined to the CA Top Secret Security File. You no longer need the UADS conversion program. Add all subsequent new TSO users directly to the CA Top Secret Security File, avoiding UADS completely.

Removing TSO Attributes

All TSO attributes can be removed with the following command:

```
TSS REMOVE(acid) DATA(TSO)
```

In addition, any PERMIT(s) the ACID has for the following four TSO classes must be revoked using the TSS REVOKE command.

```
TSOAUTH
TSOPROC
TSOACCT
TSOPRFG
```

Default TSO Data

In its role of acting as a UADS replacement, CA Top Secret keeps TSO-specific data pertaining to user logon defaults on the Security File. These items were formerly kept on the user's UADS member. These items can be updated through the TSS command (the administrator must have MISC2(TSO) authority), and most can also be changed by the user himself when logging on. When the user logs on, TSO displays the logon panel with the current settings of each of these items; at that time, the user is free to change any item. Changes are recorded on the Security File and appear on the logon screen the next time that user logs on.

TSO issues a RACXTRT during logon to extract default TSO logon data from the Security File. If none of the requested fields exist for the user, TSO will attempt to locate the user's record in SYS1.UADS.

If any of the TSO data fields are defined for the user, TSO will not interrogate SYS1.UADS. If any required logon fields are not defined in the user's Security Record, TSO will prompt for the missing information from the TSO logon screen.

Note: No TSS administration can be performed on the TRBA and the TRBA fields, which are displayed when listing a user's TSO data. These fields display the following information:

TRBA

Relative block address (RBA) of the user's mail directory entry in the broadcast data set

TUPT

Value of the user profile table (UPT)

Note: TSO logon remains under the control of CA Top Secret and UADS information will be ignored until every attribute and TSO permit defined to CA Top Secret for the ACID is removed or revoked. To restore an ACID to UADS control you must remove all of the 16 attributes that could possibly exist for the ACID using the TSS REMOVE command. The attributes are:

```
TSOLPROC() TSOLACCT() TSODEFPRFG() TSOCOMMAND()
TSOUNIT() TSOHCLASS() TSOJCLASS() TSOMCLASS() TSOSCLASS()
TSOUPT() TSOLSIZE() TSOMSIZE() TSOUDATA() TSOOPT() TSORBA() TSODEST()
```

TSO Keywords

A list of the TSS keywords used with TSO default data fields and a brief description appear next. The TSS ADD/REMOVE and REPLACE command functions are used to accommodate these items in the CA Top Secret environment.

TSOLACCT

The one- to 40-character logon account number. Account numbers are also TSO-related resources and the user must be permitted to any account number with which he attempts to log.

TSOCOMMAND

The one- to 80-character logon command. This command is executed by TSO immediately after logon (assuming successful completion). The logon command is not required and the user need not fill it in at logon time if blank. The user must have the appropriate security permissions (that is, PROGRAM, LCF) as warranted for the command chosen. For example, if TSOCOMMAND(TEST) is chosen, the userid must have PROGRAM and LCF authority to the TSO TEST command if they would normally be required. Using TSOCOMMAND does not exempt the user from normal security checking.

TSODEST

The one- to eight-character destination name for the session output. It is not required for, and cannot be changed at, logon.

TSOHCLASS

The one-character hold class. It is not required for, and cannot be changed at, logon.

TSOJCLASS

The one-character job class. It is not required for, and cannot be changed at, logon.

TSOMCLASS

The one-character message class. It is not required for, and cannot be changed at, logon.

TSOSCLASS

The one-character sysout class. It is not required for, and cannot be changed at, logon.

TSOLPROC

The one- to eight-character logon procedure name. Procedure names are also TSO-related resources and the user must be permitted to any procedure name with which he attempts to log on. Also, the user must specify a procedure name to log on, and have security permissions to any data sets allocated by that chosen procedure name. If this value is not set with a TSS command, he must fill in this field on the screen at logon time.

TSOLSIZE

The one- to seven-numeric character region size at logon. The value chosen denotes the region size, in kilobytes, and can be changed at logon.

TSOMSIZE

The one- to seven-numeric character representing the maximum region size, in kilobytes, which can be specified at logon by the user. (That is, the region size at logon must be less than or equal to this value.)

TSOOPT

The logon options that include these values: MAIL/NOMAIL, NOTICES/NONOTICES, and OIDCARD/NOOIDCARD. These values may be coded on a TSS command; the user can change the options by menu selections at logon time.

TSODEFPRFG

The one- to three-numeric character representing the performance group number, whose value can range from 1 to 255. Performance groups are used by the system to determine allocation of system resources (for example, CPU time) based on installation-defined criteria. Performance group numbers are also TSO-related resources and the user must be permitted to any group number with which he attempts to log on. A performance group number is not required, and cannot be changed at logon.

TSOUDATA

The four-character installation-defined data whose values range from 0 to 9, and A to F. This data is used by installations for any chosen purpose. It is not alterable by the user at logon time and not required for, and cannot be changed at logon.

TSOUNIT

The one- to eight-character unit (device) name for dynamically-allocated data sets. The name must be a defined generic unit class name at the installation. This field is not alterable by the user at logon and is not required for successful logon.

TSO-Related Resources

There are four predefined resource classes to support the CA Top Secret UADS replacement:

TSOACCT

(account numbers)

TSOPROC

(JCL procedure names)

TSOPRFG

(performance group numbers)

TSOAUTH

(TSO authorities/privileges)

A list of the TSS keywords for the TSO-related resources and a brief description appear next. The TSS ADD/REMOVE and PERMIT/REVOKE command functions are used to implement these resources within the CA Top Secret environment.

TSOACCT

All one- to 40-character account numbers must be owned and permitted to users wishing to use a specific number. TSO issues a security request on the account number specified at logon; if this check fails, TSO prompts the user for a new account number. Logon does not complete unless the user is authorized to an account number. Ownership and permission are required. The TSS ADD and PERMIT commands can be used to assign ownership and authorization, respectively, to TSOACCT resources.

TSOPROC

All one- to eight-character procedure names must be owned and permitted to users wishing to use them. TSO issues a security request on the procedure name specified at logon; if this check fails, TSO will prompt the user for a new name. Logon will not complete unless the user is authorized to a procedure name. The user must also be permitted to all data sets in the specified procedure with the TSS commands. Authorization to a procedure name does not convey authorization to the data sets in the procedure.

TSOPRFG

All one- to three-character performance group numbers must be owned and permitted to those wishing to use them. TSO will issue a security request on the group number specified at logon; if this check fails, TSO will prompt the user for a new number. The user can complete logon by specifying a new number or blanking out the performance group number field on the logon screen.

TSOAUTH

In the UADS environment, these authorities are known as attributes. They specify whether or not a given user has the authorities or privileges that are implied by their assignment. Attributes are as follows:

- ACCT-Account authority; required to use the ACCOUNT command (to maintain UADS) or the SYNC command.
- JCL-Submit authority; required if the user is to submit batch jobs.
- OPER-Operator authority; required if the user is to use the TSO CANCEL/DISPLAY commands.
- MOUNT-Required for mount capability.
- RECOVER-Required if the user is to use edit recovery.
- CONSOLE-Required if the user is to use the TSO CONSOLE command to implement extended MCS consoles. This resource requires the use of the TSS UADS replacement feature.
- PARMLIB-Required if the user is to use the TSO PARMLIB command.

TSO issues a security request on these authorities at logon and saves the results in an internal table. When the need arises, TSO refers to the internal table rather than doing another security request. For example, if the user tries to submit a job, TSO checks to see if the security request for TSOAUTH(JCL) was successful at logon. If not, the user is denied the job submission. The user may subsequently be permitted to TSOAUTH(JCL), but he then must log off and log back on to have TSO become aware of it.

These seven TSOAUTH resources must be owned and authorized to those users that require them. Again, ownership is required; if one or more is not owned, the security call will set a return code of 4 to TSO which interprets this as a denial of access. For example, if TSOAUTH(JCL) is not owned, no user (except those whose IDs are obtained from UADS) is able to submit jobs. The spelling of these resources is critical for the same reason. TSOAUTH(ACCT) is correct, while TSOAUTH(ACCOUNT) is not an acceptable substitute.

Again, it should be stated that TSO must find at least one of the fields described in 4.1, “Default TSO Data” from the Security File or else it will interrogate UADS for all data. This means that TSO will determine authorization to ACCOUNT, SUBMIT, and so on, from the UADS attribute settings, rather than from security requests. However, if TSO does find at least one field on the Security File, it ignores UADS for the duration of the session.

Synchronization With SYS1.BRODCAST

CA Top Secret supports the SYNC command (or the equivalent SYNC subcommand of ACCOUNT) which updates the SYS1.BRODCAST directory by adding new TSO users and removing deleted IDs from it. Any CA Top Secret ACID which has TSO default data attached, will be added to the SYS1.BRODCAST directory when SYNC with the RACF or BOTH option is used. The ID under which SYNC is run must have ACCOUNT authority, which is a TSO requirement.

It is not necessary to maintain SYS1.UADS and TSS in order to keep in sync with SYS1.BROADCAST. You can issue TSO SYNC RACF to synchronize what is in TSS with what is in SYS1.BRODCAST. You can issue TSO SYNC UADS to synchronize what is in UADS with what is in SYS1.BRODCAST. And you can issue TSO SYNC BOTH to synchronize what is in both UADS and TSS with what is in SYS1.BRODCAST.

CA Top Secret UADS Conversion Program

CA Top Secret provides a conversion program to facilitate the insertion of TSO data into the CA Top Secret Security File. TSSCVUAD provides the following functions:

- Reads the UADS data set and constructs TSS commands to create user IDs, assign default TSO fields to user IDs, assign ownership of procedure names, account numbers, performance group numbers and authorities, and to permit them to user IDs as appropriate.
- Commands are written to a disk file which can be edited through ISPF EDIT; the changed file can then be input to CA Top Secret in batch to install the TSO users onto the Security File. An optional user exit can be called for each new userid, procedure name, account number and performance group number encountered. The exit can decline conversion or assign ownership of a user ID, procedure name, account number, or performance group to a specific department instead of the default (TSODEPT1).
- A user profile table, broadcast RBA, and installation data will be installed if these items exist for a given user.
- Ownerships and permissions for all procedure names, account numbers, and performance groups assigned to a given user are converted.
- A list of member names converted and skipped, as well as the count of each, is printed by the program.

TSSCVUAD JCL

To run the conversion program (TSSCVUAD), use the JCL given below. Lowercase bold type must be replaced with the appropriate parameters for your installation.

```
//jobname    JOB

                   UADS Conversion Program (TSSCVUAD)

Instructions:
1. Insert a valid jobcard (above).
2. Tailor the parameters to conform with your installation standards.
3. Submit this job.

[]/CONVERT    EXEC PGM=TSSCVUAD [,PARM=NOCRE,]OFAC
//STEPLIB    DD   DSN=tss load library,DISP=SHR
//UADSIN     DD   DSN=sys1.uads,DISP=SHR
//SYSPRINT   DD   SYSOUT=*
//SYSABEND   DD   SYSOUT=*
//CMDOUT     DD   DSN=dca01.uads.cmds,DISP=(,CATLG),
//                UNIT=3380,VOL=SER=nnnnnn,
//                SPACE=(TRK,(1,1)),DCB=BLKSIZE=nnnn
/*
```

Parameter Values

NOCRE

Suppresses the TSS CREATE(acid) command for each USERID in SYS1.UADS. It is an optional parameter value.

NOFAC

Suppresses the generation of a TSS ADDTO(acid) FACILITY(TSO) for each USERID in SYS1.UADS. It is an optional parameter value.

DD Statements

STEPLIB

Defines the library containing TSSCVUAD and the optional user exit module.

UADSIN

Defines the DDNAME pointing to your UADS data set.

SYSPRINT

Defines the spool file for the member list report.

CMDOUT

Defines the output file containing the TSS commands created by the program. The record format is VB and the LRECL is 255.

Optional User Exit for TSSCVUAD

The optional user exit module should be compiled and linked with the name TSSUADEX. It should reside either on the STEPLIB for the conversion job or in the linklist. TSSCVUAD will LOAD this module when it starts up; if the exit is missing, TSSCVUAD proceeds without it.

The user exit is called with the following conditions in effect:

- R1 points to a list of address words as described below
- R13 points to an 18 word save area; the exit should save TSSCVUAD's registers here
- R14 is the return address to TSSCVUAD
- R15 is the address of the exit routine

The word list pointed to by R1 consists of three fullwords:

- F'index-value'
- A(resource-name)
- A(department-name)

The index value indicates why the exit was called and what the other parameters will be:

4

Start of conversion of new userid. The second word of the list is the address of the userid (eight characters, left-justified, padded with blanks) and the third word points to an eight-byte blank field. The exit should set return code 4 to cause TSSCVUAD to skip conversion of this userid, or return code 0 to allow it to proceed. If return code 0 is set, the user exit may specify the name of the department to which the userid is to be added, by placing this name in the field pointed to by the third word of the PLIST. The default department is TSODEPT1.

8

New procedure name encountered. The second word points to a 40-byte field containing the procedure name (left-justified, padded with blanks) and the third word points to an eight-byte field where the exit may specify the name of the department that is to own this procedure name. The default is TSODEPT1. Return codes are ignored for this call.

12

New account number encountered. The second word points to an eight-byte field containing the account number (left-justified, padded with blanks) and the third word points to an eight-byte field where the exit may specify the name of the department that is to own this account number. The default is TSODEPT1. Return codes are ignored for this call.

16

New performance group number. The second word points to the three-digit performance group number in EBCDIC, (right-justified, padded with zeros) and the third word points to an eight-byte field where the exit may specify the name of the department that is to own this group number. The default is TSODEPT1. Return codes are ignored for this call.

When the user exit has completed processing, it should return to TSSCVUAD. It does this by reloading the registers from the save area, placing the return code in register 15, and branching to register 14.

CA Top Secret Panel Use

The CA Top Secret Administration panels are written in C language and do not run as a typical ISPF Dialog Manager Application.

Administration and Simulation Menu Installation

The load module TSSISPF calls the CA C-Runtime r3.1 Library (FMID SF33100), part of the CA Common Services for z/OS. To use the Administration panels:

- For normal administration, set the ISPF menu to call the CLIST TSSISPF (...'COMMAND(%TSSISPF)').
- For control option administration, set the ISPF menu to call the CLIST TSSISPFM (...'COMMAND(%TSSISPFM)').

  The CAI.CAIPNL0 library contains examples of the ISPF main menu panel for various ISPF releases in members named ISR@V*x*R*x* where V*x*R*x* is the ISPF version and release number.
- Copy the CLISTs TSSISPF and TSSISPFM into your SYSPROC library concatenation from CAI.SAMPJCL. (These are called using a CLIST because they do not preserve the IS environment.)
- Make the C-RUNTIME library, and the CA Top Secret library containing the TSSISPF module, available to TSO/ISPF users (as ISPLLIB or STEPLIB in the TSO procedure).

The CA Top Secret Simulator (TSSSIM) panels are written as an ISPF Dialog Manager Application. The load module TSSSIM is invoked from ISPF as follows:

- Concatenate the following libraries to the ISPPLIB libraries:
  - CAIMSG0
  - CAIPNL0
  - CAK0PROF
  - CAITBL0

  or
  - Copy members CANO01, SIM40, and SIM41 from the CAIMSG0 library into an existing ISPMLIB library.
  - Copy all members from the CAIPNL0 library into an existing ISPPLIB library.
  - Copy CAK0CMDS from the CAITBL0 library into an ISPTLIB library.
- Copy CAK0PROF into an ISPPROF library.

TSS Simulation will now be available within ISPF by entering "SS" in the primary options menu.

The Split Screen Option

To use the TSO/ISPF split screen feature with CA Top Secret administration panels, add:

```
SELECT COMMAND(%TSSISPFS) NEWAPPL(TSS)
```

wherever TSSISPFS is being called (TSO command, menu option, or CLIST).

Split Screen with Administration Panels

To use the TSO/ISPF split screen feature with the CA Top Secret administration panels, you must have:

- A PROFILE dataset containing the CA C-Runtime Profile allocated to your TSO/ISPF environment. This dataset is added to your normal PROFILE concatenation and contain the member CACCENV distributed with the CA C-Runtime component of Common Services. If you are running any CA ViewPoint component, a PROFILE dataset has already been allocated.
- The ISPPLIB dataset containing the CA C-Runtime Panel Models allocated to your TSO/ISPF environment

To use the TSO/ISPF split screen feature

1. (Optional) Create a partitioned dataset (PDS) (RECFM=FB,LRECL=80) with a single member, CACCENV, in it. This member must contain one statement with the text, "SCR3270=SPF" in column 1. This dataset must be allocated to the PROFILE DD in your ISPF environment.
2. Allocate the ISPPLIB dataset containing the CA C-Runtime Panel Models to your TSO/ISPF environment. This dataset is added to your normal ISPPLIB concatenation. The dataset must contain members CACPNLM2, CACPNLM3, and CACPNLM4 found in the file TSSISPFM.
3. Copy and customize to site specifications the CLIST TSSISPFS, found in the file CAI.SAMPJCL.
4. The CLIST contains allocate statements for the PROFILE and ISPPLIB files. You can:
   - Customize these to point to your datasets
   - Pre-allocate the datasets and remove the allocations from the CLIST

The panels are invoked by executing the CLIST, TSSISPFS. The CLIST reserves function keys 1 through 4 and 7 through 8 as "undefined" to make them available to the CA C-Runtime. These keys are restored to their original settings when TSSISPFS ends.

Chapter 7: Implementing Security for CA IDMS

This section contains the following topics:

Installation of CA IDMS

The CA Top Secret CA IDMS Interface is invoked using the CA Common Services for z/OS component. This is done automatically when external security is specified in the CA IDMS Signon Resource Types Table (#SRTT).

For information on the #SRTT, see the CA *IDMS Security Administrator Guide*.

- The CA IDMS signon, interface includes a site-defined secondary resource check. An appropriate resource class must be defined to support this. Program definitions are also required for the TSS command task and the application interface. Therefore, the steps required to install security for the CA IDMS are:
- Define the CA IDMS #SRTT
- Define the signon resource class
- Install the TSS Command Task
- Install the application interface

Define the CA IDMS #SRTT

For information on defining the CA IDMS #SRTT, see the CA *IDMS Security Administrator Guide*.

The following CA IDMS security macros cause CA IDMS to invoke CA Top Secret for signon processing and task checking. Additional entries can be specified to protect additional resources.

```
#SECRTT TYPE=INITIAL,
                SVCNUM=nnn,
                ENVNAME=envname,
                SGNRETN=0

#SECRTT TYPE=ENTRY,
               RESTYPE=SGON,
                SECBY=EXTERNAL,
                EXTCLS=IDMSSYS,
                EXTNAME=(RESTYPE,RESNAME)

#SECRTT TYPE=ENTRY,
                RESTYPE=TASK,
                SECBY=EXTERNAL,
                EXTCLS='LCF',
                EXTNAME=(RESNAME)

#SECRTT TYPE=OCCUR,
                RESTYPE=TASK,
                SECBY=OFF,
                RESNAME='TSS'

#SECRTT TYPE=OCCUR,
                RESTYPE=TASK,
                SECBY=OFF,
                RESNAME='SIGNON'

#SECRTT TYPE=OCCUR,
                RESTYPE=TASK,
                SECBY=OFF,
                RESNAME='SIGNOFF'

#SECRTT TYPE=OCCUR,
                RESTYPE=TASK,
                SECBY=OFF,
                RESNAME='BYE'

#SECRTT TYPE=FINAL
```

Note the following:

- Specifying EXTCLS='LCF' allows you to define the task to CA Top Secret using either LCF or OTRAN.

- If both LCF and OTRAN definitions exist for a given transaction, then OTRAN checking will occur.
- When implementing database security in CA IDMS, remember that activation of external security for RESTYPE=DB (database) will also activate security for resources: ARE, NRU, QSCH, NSCH, DACC, TABL, DMCL, and DBTB. Appropriate #SRTT entries must be coded for these resource classes, as well as RESTYPE=DB.

Modify this table with the IDMS.SAMPJCL member UMODSRTT, created during the IDMS installation steps. It is the user's responsibility to maintain the SMP/E usermod associated with this assembly.

Define the CA IDMS Signon Resource Class

CA IDMS drives an additional resource check at signon. The resource class and resource name are determined by the values coded in the #SRTT entries shown in the section Defining the CA IDMS #SRTT. The resource check is a simple check of permissions-no access levels are specified.

Using the previous #SRTT example, a system resource class would need to be added to the RDT. The following sample TSS command demonstrates how this would be done:

```
TSS ADD(RDT) RESCLASS(idmssys)
             RESCODE(xx)
             ATTR(LIB,PRIVPGM,DEFPROT)
             MAXLEN(36)
             ACLST(NONE,ALL)
             DEFACC(ALL)
```

The RESCODE value *xx* should be a previously unassigned value for a user-defined resource class.

The external name specified for the IDMS #SRTT SGON resource class is coded as *EXTNAME=idmssys*. The IDMS SGON resource is associated with the CA Top Secret RESCLASS *idmssys*. CA IDMS will use the IDMS sysgen SYSTEM name as the resource name. The complete CA Top Secret resource name for signing on to system IDMS120 with #SRTT is *idmssys* (SGON.IDMS120). CA IDMS will fail all signons if the resource class is not owned and permitted to the users.

The following commands show how to add IDMS120 to the *idmssys* resource class in the department idmsdept and then PERMIT the user identified by usracid to access the signon resource.

```
TSS ADD(idmsdept) idmssys(SGON.)

TSS PERMIT(usracid) idmssys(SGON.IDMS120)
```

Replace system IDMS12 with the name of the IDMS system parameter being used to initialize your IDMS region.

Install the TSS Command Task

Use this procedure to install the TSS command task.

To install the TSS command task

1. To assure that the TSSIDMS program is accessible in the IDMS application library, do *one* of the following:
   - Copy the load module TSSIDMS from the CA Top Secret load library into the CA IDMS program load library.
   - Concatenate the CA Top Secret CAILIB into the CDMSLIB application load library.
2. Define the program TSSIDMS in the CA IDMS system generation, as follows:
   ```
   PROGRAM TSSIDMS LANGUAGE IS ASSEMBLER NOPROTECT
   ```
3. Define the task TSS in the CA IDMS system, as follows:
   ```
   TASK TSS INVOKES PROGRAM TSSIDMS
   ```

Install the Application Interface

Use this procedure to install the application interface.

To install the application interface

1. To ensure that the TSSMAI program is accessible in the IDMS application library, do *one* of the following:
   - Copy the load module TSSMAI from the CA Top Secret load library into the CA IDMS program load library.
   - Concatenate the CA Top Secret CAILIB into the CDMSLIB application load library.
2. Define the program TSSMAI in the CA IDMS system generation, as follows:
   ```
   PROGRAM TSSMAI LANGUAGE IS ASSEMBLER NOPROTECT
   ```

Note: Do not specify the NOSAVEAREA parameter.

Multi-User Address Space

Under CA IDMS, each user signed on to the region occupies part of the CA IDMS address space. Consequently, when a user requests access to a resource, the operating system "sees" CA IDMS performing the access-not the individual user. This process is typical of multi-user address spaces in general (like CICS). Therefore, to verify resource accesses on the individual user level, CA IDMS must issue its own security checks on behalf of the user performing the resource access. Since CA IDMS performs these checks for all users within the address space, there is no reason for MVS to perform a duplicate check for CA IDMS.

However, you should note that this resource checking is performed by the CA IDMS External Security Manager.

CA IDMS controls whether the same user signs on more than once by using the MULTIPLE SIGNON parameter. If this parameter is set to YES, a user can sign on to multiple terminals. Each signon after the first one shares the ACEE from the first terminal the user signed on to; therefore, subsequent signons will not be traced.

Since there is only one signon from the viewpoint of CA Top Secret, setting SIGN(S) on the facility will not disable the feature. To refresh a user's security environment when using MULTIPLE SIGNON, the user must sign off of all terminals before signing back on.

The NOXDEF and XDEF Suboptions

The NOXDEF FACILITY suboption is set by default to allow all users to execute any task until access controls have been established by either an inclusive or exclusive list for the user. To provide default protection for a task, set the XDEF FACILITY suboption.

The XDEF suboption indicates that users must have some kind of task list- either TRANS or XTRANS, before the user can execute any task.

Administration Requirements

The TSS command can be used through CA IDMS for all administration and auditing functions. All changes made via the TSS command in any facility are effective immediately across all CPUs that share the Security File. If the change is made to a user already signed on, the user must simply sign off and sign on again to retrieve the updated security record.

CA IDMS Definition

CA IDMS can be defined to CA Top Secret as either a started task or a batch.

CA IDMS as a Started Task

CA IDMS can be defined to CA Top Secret as a started task through the operations detailed in this section. Defining a started task to CA Top Secret results in the association of that STC with a specified ACID.

```
TSS CREATE(IDMSP) TYPE(USER)
                  NAME ('PRODUCTION IDMS')
                  DEPT(dept) FAC(STC)
                  MASTFAC(IDMSPROD)
                  PASS(NOPW,0)
```

- The ACID that is associated with the started task must be created with FAC(STC) to allow execution as a started task.
- The department must already exist.
- Protected data sets, and other resources that are to be accessed directly by the CA IDMS region, must be permitted to the ACID.
- If batch jobs are submitted from the region, attach the NOSUBCHK attribute to the Control ACID.

After the ACID associated with the started task has been created, the ACID must be added to the STC record through a TSS ADD entry. For example:

```
TSS ADD(STC) PROC(IDMSPR)
             ACID(IDMSP)
```

Note: This example assumes that the CA IDMS PROC name is IDMSPR. Any name can be used.

CA IDMS as a Batch Job

CA IDMS can also be defined to CA Top Secret as a batch job through the following operation:

```
TSS CREATE(IDMSP) NAME('PRODUCTION IDMS')
                  TYPE(USER)
                  DEPT(dept)
                  FAC(BATCH)
                  MASTFAC(IDMSPROD)
                  PASS(NOPW,0)
```

- The ACID is created with FAC(BATCH) to allow execution as a batch job.
- The department must already exist.
- Protected data sets, and other resources that are to be accessed directly by the CA IDMS region, must be permitted to the ACID.
- If batch jobs are submitted from the region, attach the NOSUBCHK attribute.
- Code the USER= parameter on the batch job statement, then submit the necessary JCL.

Modes of Operation

The following table provides a summary of access operation by mode for use in predicting system action:

Function	**Mode DORM**	**Mode WARN**	**Mode IMPL**	**Mode FAIL**
Signon Processing Required		X	X	X
Full Password Controls		2	2	X
Task Security Active		3	X	X
Program Protection Active*		3	X	X
Subschema Protection Active		3	X	X
Area Protection Active		3	X	X
#SRTT Defined Resource Protection		3	X	X
Terminal Protection Active		3	X	X
TSS Command Available	X	X	X	X
Application Interface Available	X	X	X	X
Logging Available		X	X	X

Function	**Mode DORM**	**Mode WARN**	**Mode IMPL**	**Mode FAIL**
Signon Processing Required		X	X	X
Auditing Available		X	X	X

X

All users in all conditions.

1

If explicit signon is performed.

2

For defined users only.

3

Messages and logging only.

Note the following:

- When securing CA IDMS, CA Top Secret only protects those resources that have corresponding #SRTT entries.
- Security Control Administrators (MSCAs, SCAs, LSCAs, ZCAs, VCAs, DCAs) can use their CA Top Secret password in any mode.

Logging Options

Full logging support is available in the CA IDMS environment. Logging can be specified on a global basis via the LOG control option, or on a facility basis via the suboption of the FACILITY control option. Details regarding the operation and assignment of logging can be found in the *Control Options Guide.* A summary of logging options appears in the following table for quick reference.

Logging Option	**Function**
ACTIVITY	Logs all activity
INIT	Logs all signon and signoff activity
ACCESS	Logs all resource accesses
ALL	Selects all log options
NONE	Deactivates logging
MSG	Requests writing of all violation messages to a user's terminal
SMF	Requests writing of data to SMF data sets

Sec9	Requests violation messages be routed to security console (route code 9)

It should be noted that certain options will generate large amounts of data, specifically ALL and ACTIVITY. If these options are selected, verify that enough space has been allocated in the Audit tracking file to prevent loss of data. SMF reporting is especially vulnerable, due to global implications of a "data lost" situation affecting system performance and monitoring. For this reason, and for ease of reporting, the CA Top Secret Audit/Tracking File should be used for all violation and auditing purposes.

Auditing Options

Use CA Top Secret logging and auditing mechanisms to easily audit all defined users and ownable resources.

For defined users, add the AUDIT attribute to the ACID. For example:

```
TSS ADD(USER'5) AUDIT
```

For ownable resources (programs, subschemas, areas, and terminals) add the resource name or prefix to the AUDIT record, as shown below.

```
TSS ADD(AUDIT) AREA(TSTPAY)
```

You can access audited resources from all modes.

Implementing Security

The CA Top Secret CA IDMS Interface is designed as a comprehensive CA IDMS security system. CA Top Secret CA IDMS security support provides extended function in the areas of:

- Signon control
- Task security
- Program security
- Database security
- #SRTT defined resource security
- Facility (region) control
- Online security administration through CA IDMS
- CA IDMS Application Interface to CA Top Secret
- Job submission control

Environment Support

The CA Top Secret CA IDMS Interface provides extended resource control within the CA IDMS environment, in addition to the existing CA IDMS security features.

CA Top Secret security is supported for all externally protected resources.

Signon Security

As a user-oriented security product, CA Top Secret requires that all users identify themselves to CA Top Secret to use CA IDMS. This can be done with a user-initiated signon or by using the Automatic Terminal Signon feature.

This release requires that all users must be permitted to the signon resources designated in the #SRTT.

- Comprehensive password controls are available, including:
- Password masking
- Minimum password length
- Suppression of vowels
- Random generation of passwords
- No password (NOPW option)
- Automatic password expiration on a pre-defined interval from 0 (no password changes required) to 255 days
- Minimum number of days between password change
- Password history comparison
- Restricted password prefixes (RPW option)
- Name and signon ID comparison
- Automatic ACID suspension on an installation defined threshold of number of failed attempts (default is three).

SIGNON Task

Most users will have to explicitly sign on to CA IDMS, using the SIGNON task. The format of the SIGNON task is:

```
SIGNON acidname password/new-pswd
```

acidname

The user's one-to-eight character ACID; required.

password

The user's password; always required unless the user has a password of NOPW.

new-pswd

Specifies a new password; optional. Specifying RANDOM causes CA Top Secret to automatically generate a new random password for the user.

Automatic Terminal Signon

CA Top Secret allows for the automatic signon of an CA IDMS terminal. This feature is useful for applications requiring minimal security, or where the physical security surrounding a terminal is adequate for the sensitivity of the application. Automatic Terminal Signon is also useful for receive-only terminals, which are used for displaying secured information.

Automatic Terminal Signon is performed whenever a task is entered prior to performing an CA IDMS signon, and there exists a CA Top Secret ACID by the same name as the terminal name.

The terminal name used will depend on your environment:

- For a VTAM terminal, the VTAM node name will be used.
- For all other terminals, the CA IDMS PTERM will be used.

Terminals with ACIDs defined will automatically be eligible for Automatic Terminal Signon. If there is no ACID defined with the same name as the terminal defined to CA Top Secret then the task will be failed and the user will receive an CA IDMS message requesting that an explicit signon be performed.

If the ACID exists, then CA Top Secret performs the security checks (4) through (9), described previously, for security checking. If these checks are successful, the ACID is associated with that terminal until signoff, just as if an explicit signon had been performed. Finally, the task will be processed (subject to LCF task security).

Note: If the Automatic Terminal Signon is successful, the user who entered the task will not be aware of the process.

Several special requirements must be fulfilled before Automatic Terminal Signon (ATS) is supported:

- The SRTT TYPE=INITIAL entry must be assembled with the value DFLTSGN=YES.
- This entry suppresses default ACID signon (explained below).
- The SRTT TYPE=ENTRY for RESTYPE=TASK must be set to SECBY=EXTERNAL.
- Tasks designated as "safe" never invoke ATS. See the section OTRAN Security for information on designating a task as safe.

Using Default ACID Signon

CA IDMS makes the following special requirement when using default ACID signon:

- The SRTT TYPE=INITIAL entry must be assembled with the value DFLTSGN=NO.
- In CA IDMS, Automatic Terminal Signon and default ACID signon are mutually exclusive, since the above SRTT entry suppresses Automatic Terminal Signon.

Then the security administrator:

- Sets a facility DEFACID for any CA IDMS multi-user facility. An example follows:
  ```
  TSS MODI FAC(IDMSPROD=DEFACID(dfltacid))
  ```
- Must explicitly create the *dfltacid* specified for the facility. It is recommended that the ACID be of limited access since its use is automatic.
- If the end-user attempts to sign on to the facility with a blank or non-existent ACID, the *dfltacid* will be used to sign the user on to the facility.

The security checks listed in (4) through (9) in section 1.8.2.1, "SIGNON Task," are performed on the default ACID signon.

Administering Task Security

CA Top Secret gives you two options for protecting CA IDMS tasks. You can protect them on the resource level as owned resources through the resource name OTRAN (Owned TRANsaction) or on a user-by-facility basis through the use of the Limited Command Facility (LCF).

To protect tasks, an appropriate entry must be made in the #SRTT to cause CA IDMS to drive external security. You have three options for implementing task security by coding either EXTCLS='LCF', EXTCLS='LCFONLY', or EXTCLS='OTRAN' in the #SRTT entry for RESTYPE=CLASS. An example of the CA-IDMS macros used to protect tasks appears below.

```
#SECRTT TYPE=ENTRY,
RESTYPE=TASK,
SECBY=EXTERNAL,
EXTCLS='LCF',
EXTNAME=(RESNAME)
```

For the value EXTCLS='LCF', CA Top Secret tests task security as follows:

- If the task is defined through ownership of an OTRAN resource, determine if EXEC access is permitted to the task.
- If the task isn't defined through OTRAN ownership, test for LCF TRANS and XTRANS rules, and then determine if access is permitted.
- If the CA-IDMS facility XDEF control option is set and neither OTRAN or LCF rules have rejected the access, then reject the access.
- If the CA-IDMS facility NOXDEF control option is set, allow the access.

Note: CA Top Secret automatically checks for OTRAN and LCF permissions for EXTCLS='LCF'.

For an explanation of the LCF, OTRAN, and LCFONLY values, see the *User Guide*.

OTRAN Security

The OTRAN resource name is shared by all CA IDMS, IMS, and CICS facilities. Therefore, protecting a task via OTRAN for a CICS region also results in tasks of the same name being protected in all CA IDMS, IMS, and CICS regions which are also under the control of CA Top Secret.

Note: A task protected via OTRAN does not go through LCF checking unless EXTCLS='LCFONLY' is specified.

To add ownership of a task to an ACID, enter:

```
TSS ADD(acid) OTRAN(task)
```

To allow users access to the task, enter:

```
TSS PERMIT(acid) OTRAN(task)
```

Providing Safe Tasks

The #SRTT allows you to indicate that certain tasks should bypass security checking. This is particularly useful for customized signon programs or menus. The following sample entry demonstrates how to bypass security for the SGN2 task:

```
#SECRTT TYPE=OCCUR
RESTYPE=TASK,
SECBY=OFF,
RESNAME=SGN2,
EXTCLS='LCF' EXTNAME=(RESNAME)
```

LCF Security

If you choose not to protect tasks using OTRAN, they can be protected via LCF. Tasks protected through LCF must be defined by facility. Tasks can be defined either inclusively (TRANS) or exclusively (XTRANS), but not both. Essentially, each user can have an inclusive list, which specifies a list of tasks the user is allowed, or an exclusive list which the user is not allowed to use.

It is recommended that tasks be divided by function or subset and defined as a group within profiles. This way tasks are defined only once per group, instead of once per user.

CA Top Secret provides protection for all resources that are coded in the #SRTT. CA IDMS fails all resource checks for undefined resources.

Since CA IDMS does not tolerate security violations during abend processing, care must be taken when protecting certain resources. For example, if you activate external security for SPGM and SLOD and the program resources beginning with IDMS and RHDC are unowned, the CA IDMS clean-up tasks CLOD QUED abend in any mode. If these programs are owned and the CA IDMS started task ACID has not been permitted to use them, CA Top Secret issues warning messages while in WARN mode, but the transactions will still ABEND.

For resources to be protected they must be owned. The standard set of CA Top Secret control features for owned resources is also available for CA IDMS resources including:

- Generic prefixing
- Facility (region) restriction
- Expiration, day of week, and time of day limiting
- ACTION control (specifies which CA Top Secret action takes effect if rules in the PERMIT function are enforced)
- Program pathing
- I/O access levels

It is important to remember that an ownable resource (not a task) can be added to the Audit Record so that all activity involving it can be tracked. In short, resource ownership allows more comprehensive, flexible, and systematic security than a task-based implementation.

Since the two approaches are not mutually exclusive, some combination of the two approaches might suit your particular environment and security needs best.

Administering Program Security

Program protection is activated by coding an #SRTT entry for the SPGM resource class.

A sample #SRTT entry appears below.

```
#SECRTT TYPE=ENTRY
RESTYPE=SPGM
SECBY=EXTERNAL
EXTCLS='PROGRAM'
EXTNAME=(resname)
```

Access to the program must be granted using the PERMIT function of the TSS command.

Access can be limited to only specific regions through the FACILITY parameter as part of the program permission. Time of day, day of week, and access expiration controls are also available.

How to establish ownership of the program and permit access to it are covered in the *User Guide*.

Administering Subschema Security

There are two methods to provide protection:

- Use traditional Subschema protection without using Run Unit protection.
- Use traditional Run Unit protection without using Subschema protection.

Note: Database protection is required for both types of protection. You must code an #SRTT macro to issue security checks when using either Subschema or Run Unit protection. The macro shown below must be used for both methods.

```
#SECRTT TYPE=ENTRY
RESTYPE=DB

SECBY=EXTERNAL
EXTCLS='DATABASE'
EXTNAME=(RESNAME)
```

Subschema Protection

Traditional subschema protection uses the following statements, which do not provide Run Unit protection, in addition to the ENTRY statements listed previously.

```
#SECRTT TYPE=ENTRY
RESTYPE=NRU
SECBY=EXTERNAL
EXTCLS='SUBSCHEMA'
EXTNAME=(SSNAME)
```

Note: Using traditional Subschema protection, the CA IDMS external security interface regards all subschema resources as protected by default, whether or not DEFPROT is an attribute of SUBSCHEMA. Thus, any attempt to use a subschema resource in an application will fail if the resource is not owned.

Run Unit Protection

Traditional Run Unit protection uses the following statements, which do not provide Subschema protection, in addition to the ENTRY statements listed previously.

```
#SECRTT TYPE=ENTRY
RESTYPE=NRU
SECBY=EXTERNAL
EXTCLS='IDMS'
EXTNAME=(RESTYPE,RESNAME,SSNAME,DBNAME)
```

If you choose to create a resource class for run units, you may want to create a general resource class for the CA IDMS resources. For example:

```
TSS ADD(RDT) RESCLASS(IDMS)
             RESCODE(NN)
             ATTR(MAXLEN(36))
             DEFACC(READ)
             ACLIST(ALL,NONE,READ,UPDATE,CREATE)
```

Then, the run unit resource can be owned by issuing:

```
TSS ADD(USERMS) IDMS(NRU.)
```

Permitted with:

```
TSS PERMIT(USERMW) IDMS(NRU.pgmname.ssname)
```

or

```
TSS PERMIT(USERMW) IDMS(NRU.pgmname.ssname.dbname)
```

Run Unit Subschema Protection

Access to the subschema must be granted using the PERMIT function of the TSS command. Access can be limited to only specific regions through the FACILITY keyword as part of the subschema permission. Time of day, day of week, and access expiration controls are also available.

The syntax for subschema security is shown below. The ADD function establishes ownership of the subschema.

```
TSS ADD(USER'3) SUBSCHEM(TSTPAY)
```

The PERMIT function grants access to the subschema.

```
TSS PERMIT(USER14) SUBSCHEM(TSTPAY)
```

The following example shows the keywords that can be used to further limit the use of a subschema via the PERMIT function:

```
TSS PERMIT(USER14) SUBSCHEM(TSTPAY)
                   FAC(IDMSTEST)
                   TIME(13,17)
                   DAYS(TUESDAY,THURSDAY)
                   FOR(14)
                   ACTION(AUDIT)
```

- Specifying FACILITY designates that the subschema can only be accessed through a particular facility, such as IDMSTEST. Omitting FACILITY implies access through any defined CA IDMS facility.
- TIME and DAY keywords are used to limit access to the subschema from 1:00 p.m. to 5:59 p.m. on Tuesdays and Thursdays.
- The FOR keyword specifies the duration of the access. (Instead of FOR, you can use UNTIL to specify duration.)
- The ACTION keyword is compatible with all other PERMIT keywords. The ACTION keyword causes any access matching this PERMIT to be audited.

Administering Area Security

Area security is provided for both logical and physical databases.

Area security requires the coding of the appropriate #SRTT macros, as shown in the following examples. The first example codes the required database security, while the second example codes area security.

```
#SECRTT TYPE=ENTRY
RESTYPE=DB
SECBY=EXTERNAL
EXTCLS='DATABASE'
EXTNAME=(RESNAME) #SECRTT
TYPE=ENTRY
RESTYPE=AREA
SECBY=EXTERNAL
EXTCLS='AREA'
EXTNAME=(RESNAME)
```

Access by utility load or print/punch for DBA operations is also checked. Internal checking is designated with DBAREAD and DBAWRITE access. External checking is performed with READ access combined with DBADMIN authority.

Access to the area must be granted using the PERMIT function of the TSS command.

Areas have access levels associated with them; the access level is derived from the function code on the DML call. The PRIVPGM (Privileged Program) keyword is also applicable for area limiting. Specifying PRIVPGM limits access to the area to a particular CA IDMS program or group of programs. Use of the area can be limited to specific regions through the FACILITY keyword, as part of the area definition. Time of day, day of week, access expiration, and action controls are also available.

Note: The area resource has been predefined in the RDT record with the LONG attribute, which supports name lengths of up to 44 characters when used with the PERMIT command function.

The syntax for area security is shown below. The ADD function establishes ownership of the area.

```
TSS ADD(USER'2) AREA(TSTPDA)
```

The PERMIT function allows update access to the area.

```
TSS PERMIT(USER13) AREA(TSTPDA)
                   ACCESS(UPDATE)
```

The following example shows the keywords that can be used to further limit the area via the PERMIT command:

```
FAC(IDMSPROD) PRIVPGM(TSTPAA45)
TIME('9,18) DAYS(MONDAY,WEDNESDAY,FRIDAY)
UNTIL(11/24/'1) ACTION(NOTIFY)
```

- Specifying FACILITY designates that the area can only be accessed through a particular facility, such as IDMSPROD. Omitting FACILITY implies access through any defined CA IDMS facility.
- Specifying PRIVPGM designates that the area can only be accessed through a particular program, TSTPAA45. Omitting PRIVPGM implies access through any program. Up to five programs can be specified for each PERMIT function.
- The TIME and DAYS keywords are used to limit access to the area from 9:00 a.m. to 6:59 p.m. on Monday, Wednesday, and Friday.
- The UNTIL keyword can be used to specify the duration of the access. (You can also use the FOR keyword to specify duration.)
- The ACTION keyword is compatible with all other PERMIT keywords. The ACTION keyword causes any access matching this PERMIT to issue a message to the security console (route code 9)

Area Security Access

The available access levels for areas and their relationship to CA IDMS DML calls are defined in the following table:

CA Top Secret Access Level	**CA IDMS Call Function**
READ	RETRIEVE
UPDATE	All other
NONE	N/A
ALL	N/A

NONE and ALL are CA Top Secret-only access levels, implying no access and total access, respectively.

The default access level is READ.

Area Security Keywords

The following example shows the keywords that can be used to further limit the area via the PERMIT command:

```
FAC(IDMSPROD) PRIVPGM(TSTPAA45)
TIME('9,18) DAYS(MONDAY,WEDNESDAY,FRIDAY)
UNTIL(11/24/'1) ACTION(NOTIFY)
```

- Specifying FACILITY designates that the area can only be accessed through a particular facility, such as IDMSPROD. Omitting FACILITY implies access through any defined CA IDMS facility.
- Specifying PRIVPGM designates that the area can only be accessed through a particular program, TSTPAA45. Omitting PRIVPGM implies access through any program. Up to five programs can be specified for each PERMIT function.
- The TIME and DAYS keywords are used to limit access to the area from 9:00 a.m. to 6:59 p.m. on Monday, Wednesday, and Friday.
- The UNTIL keyword can be used to specify the duration of the access. (You can also use the FOR keyword to specify duration.)
- The ACTION keyword is compatible with all other PERMIT keywords. The ACTION keyword causes any access matching this PERMIT to issue a message to the security console (route code 9)

Administering Terminal Security

Terminals are owned resources; therefore, auditing and distributed administration can easily be performed. Ownership of a terminal protects the resource across all defined facilities; however, access can be limited to only specific facilities through the FACILITY keyword as part of the terminal definition. Time of day, day of week, access expiration, and action controls are also available.

The terminal name defined depends on your environment:

- VTAM node name definition for all VTAM environments
- CA IDMS PTERM definition for all other environments

Note: Terminal protection applies across all facilities with like terminal names. CA IDMS PTERM definition also limits VTAM terminals for facilities such as TSO. Check the implication of terminal security as related to TSO, CICS, or any other online facilities under CA Top Secret control. To determine how to establish ownership of the terminal and permit access to it, see the *User Guide*.

Implementing Database Resources

Database resources are accessed through internal or external security. The correspondence between IDMS internal RESTYPE and external security EXTCLS resource classes is defined by the #SRTT assembly in the IDMS.SAMPJCL member UMODSRTT.

Database Resources in the #SRTT

When RESTYPE=DB is employed, the assembly provides default entries for all related database resources, both traditional navigational RESTYPE and SQL RESTYPE.

It is the user's responsibility to provide customized entries for any RESTYPE included in this group.

Define one or more resource classes in the RDT to correspond with the EXTCLS values you define in the entries. Do not use the same idmssys RESCLASS employed for system resources like SGON and TASK because the access levels used for DB RESTYPE is significantly different.

For information on the database resources specified in the #SRTT assembly, see the CA *IDMS Security Administration* Guide.

Define an RDT RESCLASS for IDMS DB RESTYPEs

The following RDT RESCLASS may be used as a model for all IDMS Database RESTYPEs:

```
TSS ADD(RDT) RESCLASS(idmsdb)
             RESCODE(xx)
             MAXLEN(44)
             ATTR(DEFPROT,PRIVPGM)
             DEFACC(READ=4000)
             ACCLVL(ALL,UPDATE=8000,READ=4000,CREATE=2000,DELETE=1000,NONE)
```

idmsdb

An arbitrary name assigned for the resource class in the RDT.

xx

An unassigned user-defined resource code.

Traditional RDT Entries with IDMS Resources

To maintain compatibility with prior releases, the navigational database resource types DATABASE and AREA can be defined as separate resource classes. SRTT entries can use existing RDT entries for DATABASE and AREA already defined to the RDT. RUNUNIT must be defined separately.

To allow administration of the DATABASE resource in the traditional manner use:

```
#SECRTT TYPE=ENTRY,
        RESTYPE=DB,SECBY=EXTERNAL,
        EXTCLS=DATABASE,EXTNAME=(RESNAME)
```

Database security can be defined using the model resource:

```
TSS ADD(dept) DATABASE(dbname)

TSS PER(user) DATABASE(dbname)
              ACC(UPDATE,READ)

#SECRTT TYPE=ENTRY,
        RESTYPE=DB,SECBY=EXTERNAL,
        EXTCLS=IDMSDB,EXTNAME=(RESTYPE,RESNAME)
```

In this case, administration proceed with the format:

```
TSS ADD(dept) IDMSDB(DB.dbname)

TSS PER(user) IDMSDB(DB.dbname)
              ACC(UPDATE,READ)
```

The IDMS #SRTT assembly provides maximum flexibility in defining the correspondence between IDMS internal resource types with CA Top Secret external RDT resource classes. It is the user's responsibility to define a consistent correspondence according to the mechanisms defined in the IDMS #SECRTT macro.

Application Interface

CA Top Secret provides a security call and verification interface for use by CA IDMS applications. The interface can be used to:

- Extract and update installation defined data
- Retrieve user information
- Perform security checks against CA IDMS or user-defined resources
- Perform logging of violations automatically
- Perform logging of user-defined information

Note: Checks against unauthorized resources does not fail the task; the application program handles the violation.

Invoking the Application Interface

CA Top Secret provides a security call and verification interface for use by CA IDMS applications. The interface can be used to:

- Extract and update installation defined data
- Retrieve user information
- Perform security checks against CA IDMS or user-defined resources
- Perform logging of violations automatically
- Perform logging of user-defined information

Note: Checks against unauthorized resources will not cause failure of the task; the application program handles the violation.

To invoke the CA Top Secret CA IDMS application interface, the application program must issue a link to program TSSMAI and pass it a request record that describes the processing to be performed by the CA Top Secret application interface.

Task Checking

The CA Top Secret application interface allows an application program to perform task or panel checking by specifying:

- TSSCLASS name of LCF or TRAN, and
- TSSRNAME containing the task or panel name.

No other fields are required.

CA Top Secret supplies a return code (in the TSSRC field) and a character return code (in the TSSCRC field) to the application program.

Update Installation Data

To update the Installation Data for the signed-on user:

- Specify a TSSCLASS name of DUFUPD
- Do not specify anything in the TSSRNAME field
- Place the updated Installation Data in the TSSRTN field

The Installation Data area cannot be extended.

User ACID Name Retrieval

The CA Top Secret Application Interface can be used to retrieve the ACID name of the signed-on user. This feature can be useful in application reporting and verification of users, by providing a guaranteed unique user identification.

The eight-character ACID name is retrieved by supplying a class name of ACIDNAME.

Upon return from the call, the ACID name can be retrieved from the TSSRTN return data area of the Request Record.

In addition to returning the ACID name when specifying ACIDNAME in the class field, CA Top Secret returns the following information for the signed-on user:

FACILITY

The eight-character name of the facility the user is signed on to

MODE

The eight-character MODE name of ACID

TYPE

The type of ACID (SCA, LSCA, ZCA, VCA, DCA, or USER)

TERMINAL

The terminal name of the user

SYSTEMID

The system name (SMF ID)

Other Checks

The CA Top Secret Application Interface can also perform various other checks by specifying special keywords in the TSSCLASS field. These checks are listed in the following table.

The checks in the following table are available only for the signed-on user.

TSSCLASS	Return Field	Returned Data Description
ACIDFULL	TSSACIDF	Full 32-character name of the ACID
DEPTNAME	TSSDEPTA	Eight-character name of the ACID's department
DEPTFULL	TSSDEPTF	Full 32-character name of the ACID's department
DIVNAME	TSSDIVA	Eight-character name of the ACID's division
DIVFULL	TSSDIVF	Full 32-character name of the ACID's division

Special Considerations

The following provides special considerations that administrators should keep in mind to ensure a thorough implementation of CA IDMS.

- Because CA IDMS is a multi-user address space system, it requires a restart of the region when applying maintenance.
- CA IDMS comes defined with two keys: KEY 8 for system use, and KEY 9 for user and system work areas. CA Top Secret requires both storage areas to be the same key. The FACILITY control option defaults to KEY=8.
- Since resource checks are now conditional, based on the parameters coded in the SRTT, considerations should be made to place control of this table with the security administrator.
- Access to CA IDMS must be granted by facility and the "signon resource". This is particularly important when adding new users.
- CA IDMS considers undefined (unowned) resources to be protected. Thus, all resources must be defined and permitted in FAIL and IMPL modes.

Chapter 8: Implementing Security for CA Roscoe

This section contains the following topics:

Install the CA Roscoe Interface

Important! Before installing the CA Roscoe interface, check the *CA Roscoe Security Administration Guide* to determine your security needs based on what types of security CA Roscoe offers in addition to the security provided by CA Top Secret.

CA Roscoe CA Top Secret support modules interface with, or replace, CA Roscoe user or security exits. Module TSSRXBEX must be link-edited with CA Roscoe programs LIBSERVE, ROSCOPY, ROSDATA, and LIBUTIL.

Note: For CA Roscoe r6.0 and above, you should **not** install CA Top Secret-supplied ACFEXIT and OUTEXIT modules. You should instead allow security to be handled through the (SAF) jesspool calls for the OUTEXIT. CA Top Secret protects these functions and provides a higher level of granularity of control.

- Certain exits are required in certain situations. The exits and when they are required are listed below.
- Sites that have defined LCF rules for protecting CA Roscoe commands and monitors must install the CMDEXIT. This exit is also required if sites wish to implement the CA Top Secret automatic terminal lock facility instead of the terminal locking facility provided by CA Roscoe.
- Sites that wish to receive the CA Top Secret security messages for data set access, rather than CA Roscoe messages, must implement DSAEXIT.

The JCL for the installation of the CA Top Secret/CA Roscoe interface resides in File 17 (TSSROSCO) of the distribution tape. There are two members:

INSTALL1

Copies exits and TSSSIM modules to the CA Roscoe loadlib and links TSSRXBEX with CA-Roscoe, ROSCOPY, ROSDATA, LIBSERVE, and LIBUTIL.

INSTALL2

Installs CMDEXIT, SUBEXIT, or ACFEXIT if user-written code is combined with CA Top Secret-supplied code.

CA Roscoe Modules

The following table outlines the CA Top Secret module names that are distributed, the CA Roscoe exit and monitor module names, and their respective functions:

CA Top Secret Module Name	CA Roscoe Module Name	Function
TSSRXCMD	CMDEXIT	Protects commands.
TSSRXSUB	SUBEXIT	Validates submitted jobs.

TSSRXDSA	DSAEXIT	Secures data set access authorization.
TSSRXBEX	BEXEXIT	Protects use of CA-Roscoe batch utilities (ROSCOPY, ROSDATA, LIBSERVE, LIBUTIL). Note that security calls through BEXEXIT have also been added to LIBUTIL.
TSSRTSS0*	RSSCTSS0	Authorizes security administration through signed on security terminal.
TSSRSIM0*	RSSCSIM0	Invokes TSSSIM

This CA Top Secret module has its corresponding CA Roscoe module name defined as an alias in CA Top Secret; therefore, you do not need to rename it.

You must activate the administration and simulation monitors at CA Roscoe startup by including the following parameters in your SYSIN file:

```
RUN=TSS(X)
RUN=SIM
```

Installing Maintenance

CA Top Secret uses SMP/E to install and apply maintenance.

Note: Once maintenance has been applied, the CA-Roscoe modules must be re-linked or copied into a library in the CA-Roscoe STEPLIB concatenation.

Facilities Matrix

As delivered, CA Top Secret comes with CA-Roscoe already defined in the Facilities Matrix. However, to use a facility not automatically recognized by CA Top Secret (like a CA-Roscoe test facility), an entry must be made in the Facilities Matrix. For more details about how to customize a facility not already defined in the Facilities Matrix, see Customizing the Interface.

The default attributes for CA-Roscoe are:

```
INITPGM=ROS    id=R  TYPE=007
ATTRIBUTES=ACTIVE,SHRPRF,ASUBM,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,RNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFCMD
ATTRIBUTES=MSGLC,NOTRACE,NOEODINIT,IJU,NODORMPW,NONPWR
MODE=WARN  DOWN=GLOBAL LOGGING=ACCESS,INIT,SMF,MSG,SEC9
UIDACID=8  LOCKTIME=000 DEFACID=*NONE*  KEY=8
MAXUSER=03000    PRFT=003
```

Do not reset the TYPE= values or these default attributes:

```
NOABEND   ASUBM    AUTHINIT   MULTIUSER    NONPWR
NOPROMPT  RES      SHRPRF     NOTSOC
```

CA Roscoe Definition as a Started Task

When defining CA Roscoe as a started task:

- Create the ACID that is to be associated with the started task with FACILITY(STC) to allow execution as a started task.

```
TSS CREATE(ROSCOE) NAME('PRODUCTION ROSCOE')
                   FACILITY(STC)
                   MASTFAC(ROSCOE)
                   PASSWORD(NOPW,0)
                   DEPARTMENT(dept)
```

- Defining a started task to CA Top Secret results in the association of that STC with a specific ACID.

```
TSS ADDTO(STC) PROCNAME(ROSTC1)
               ACID(ROSCOE)
```

- The ACID associated with the started task must include the bypass attributes shown below (so that CA Roscoe is able to bypass further security once a RACHECK has been issued) or you must allow CA Roscoe access to all required data sets and volumes that can be accessed through CA Roscoe. Attributes can be specified on the CREATE or ADDTO function.

```
TSS ADDTO(ROSCOE) NOVOLCHK,NORESCHK
```

- Allow CA Roscoe unlimited access to all data sets.

```
TSS PERMIT(ROSCOE) DSNAME(*****)
                   ACCESS(ALL)
```

- Include the CA Roscoe JCL in one of the system PROCLIBS.

CA Roscoe Definition as a Batch Job

When defining CA Roscoe as a batch job:

- Create the ACID with FACILITY(BATCH) to allow execution as a batch job.

  ```
  TSS CREATE(ROSCOE) NAME('PRODUCTION ROSCOE')
                     FACILITY(BATCH)
                     MASTFAC(ROSCOE)
                     PASSWORD(NOPW,0)
                     DEPARTMENT(dept)
  ```

- The ACID associated with the started task must include the bypass attributes shown below (so that CA Roscoe is able to bypass further security once a RACHECK has been issued) or you must allow CA Roscoe access to all required data sets and volumes that can be accessed through CA Roscoe.

  ```
  TSS ADDTO(ROSCOE) NOVOLCHK,NORESCHK
  ```

- Permit the CA Roscoe region ACID unlimited access to all data sets.

  ```
  TSS PERMIT(ROSCOE) DSNAME(*****)
                     ACCESS(ALL)
  ```

- Code a USER=ROSCOE keyword on the batch CA-Roscoe job statement.

Modes of Operation

CA Top Secret supports four separate modes of operation: DORMANT, WARN, IMPLEMENT, and FAIL. Modes are assigned at five different levels:

Global

The default for the entire CA Top Secret community. For example:

```
MODE(WARN)
```

Facility

Affects a particular facility within the community. For example:

```
FACILITY(ROSCOE=MODE=IMPL)
```

Profile

Affects a particular group of users attached to the profile. For example:

```
TSS PERMIT(PROF01) MODE(IMPL)
```

User

Affects a particular user within the community. For example:

```
TSS PERMIT(USER01) MODE(FAIL)
```

Resource

Forces a particular resource authorization to be processed in FAIL mode. For example:

```
TSS PERMIT(USER01) TERMINAL(L048T29)
                   ACTION(FAIL)
```

Note: The global level is implemented through the MODE control option, or on a facility level through the MODE= suboption of the FACILITY control option. The profile, user and resource levels are implemented through the PERMIT function of the TSS command.

Modes Defined

The following descriptions define the purpose and operation of each mode:

DORMANT

In DORMANT mode, CA Top Secret will not perform any security checking and will not generate violation messages.

Note: When running in DORMANT mode, DORMPW causes passwords to be checked by *both* CA Top Secret and CA-Roscoe. This might cause problems if the passwords are not synchronized within the two products. To prevent problems, use NODORMPW in DORMANT mode and let CA-Roscoe maintain the passwords.

WARN

In WARN mode, CA Top Secret performs full security checking but will not actually fail the user. All users must explicitly sign on or be signed on through the Automatic Terminal Signon feature. Violation messages are not generated.

IMPLEMENT

In IMPLEMENT mode, CA Top Secret provides full security for defined users and resources, but undefined users will still be allowed to sign on to CA-Roscoe. Resource security is in effect for all users. Protected resources are not accessible to undefined users unless the resource has been permitted to the ALL record for global access. This mode allows for a gradual migration of users to CA Top Secret control. This mode also generates violation messages.

FAIL

In FAIL mode, CA Top Secret requires that all users be defined to CA Top Secret. This mode generates violation messages.

Note: With the availability of four different security modes, security administrators can carefully control the pace of the implementation of security in the CA-Roscoe environment.

Violation Thresholds

Terminal locking due to excessive violations is enforced through CA Top Secret. If the VTHRESH control option is in effect and the action is CAN (for cancel) or SUS (for suspend), then CA Top Secret will lock the terminal when the threshold is exceeded. The user's only course of action is to sign off. If the action is VTHRESH(SUS), then the user is automatically suspended and is not able to sign on until unsuspended by a security administrator.

Logging Options

Full logging support is available and can be specified on a global basis through the LOG control option, or on a facility basis through the suboption of the FACILITY control option. Details regarding the operation and assignment of logging can be found in the *Control Options Guide*.

The logging options are:

ACTIVITY

Logs all activity

INIT

Logs all signon and signoff activity

ACCESS

Logs all resource accesses

ALL

Selects all Log options

NONE

Deactivates logging

MSG

Requests writing of all violation messages to a user's terminal

SEC9

Requests violation messages be routed to security console (route code 9)

ALL and ACTIVITY generate large amounts of data. If these options are selected, verify that enough space has been allocated to prevent loss of data. For this reason, and for ease of reporting, the CA Top Secret Audit/Tracking File should be used for all violation and auditing purposes.

Auditing Options

Use CA Top Secret logging and auditing mechanisms to audit all defined users and ownable resources.

For defined users, add the AUDIT attribute to the ACID. For example:

```
TSS ADDTO(USER05) AUDIT
```

For ownable resources (APPLICATION, PSB, DBD, and Terminal) add the resource name or prefix to the AUDIT record. For example:

```
TSS ADDTO(AUDIT) APPLICATION(TSTPAY)
```

You can access audited resources from all modes.

Signon Security

Users must sign on to CA-Roscoe using their key and password. A user's key also becomes his ACID. By default, CA Top Secret recognizes the first eight characters of the key as the ACID. To sign on, the user must be defined to CA Top Secret and authorized to access CA-Roscoe.

Password Changes

Password changes are supported at signon. The new password is entered on the Signon menu, after you specify the old password. All standard password security options can be used with CA-Roscoe.

Terminal Locking

CA Top Secret can be used to restrict the use of terminals to authorized users only. Also, with the optional installation of the command exit (CMDEXIT), automatic terminal locking becomes available.

Locking is enforced whenever a command 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.

Note: The CA-Roscoe TLOCK facility can also be implemented to secure unattended terminals.

Command and Monitor Security

The Limited Command Facility (LCF) can be used to restrict both commands and monitors, once the command exit (CMDEXIT) has been installed. LCF allows an inclusive list (which specifies a list of commands and monitors the user is allowed), or an exclusive list (which specifies a list of commands and monitors the user is not allowed).

An example of an inclusive list, where the user is allowed to use certain monitors, appears below.

```
TSS ADDTO(USER01) COMMAND(ROSCOE,(ZAP,EXP,UTILITY))
```

An example of an exclusive list, where the user is allowed to use all monitors except those specified, appears below.

```
TSS ADDTO(USER01) XCOMMAND(ROSCOE,(IMP))
```

Note: Use the TSS ADDTO(ALL) function to assign global LCF definitions just as you would to another user.

Data Set Security through ZAP, IMPORT, and EXPORT

The function of these monitors (ZAP, IMPORT, and EXPORT) are validated on a data set access basis. The following access level authorizations are required for the user to use the particular functions:

- ZAP with REP or SETSSI requires UPDATE; otherwise READ authorization is required.
- IMPORT requires READ authorization.
- EXPORT requires UPDATE authorization.

Note: Anytime a user accesses an MVS data set the DSAEXIT, if present, is invoked. The DSAEXIT, in turn, issues a duplicate RACHECK, which validates access to a data set for the purpose of returning CA Top Secret messages.

Job Submission Validation

For jobs submitted through CA-Roscoe using the optional submit exit (SUBEXIT), CA Top Secret provides an additional security control feature beyond the basic batch job validation. This security feature determines whether the submitter has the authority to submit the job. In other words, CA Top Secret checks whether the ACID of the signed on user is authorized to submit the ACID associated with this job through the USER= parameter on the job statement. If he is not, the job is stopped at submission before it is initiated. If the SUBEXIT is installed, the user is notified that the job submission has failed.

Note: Users should code the NOTIFY= parameter on the job statement so that they receive notification that the job has ended (or abended). See Customizing the Interface for details on customizing the SUBEXIT module.

Job Output Spool Security

CA-Roscoe r 6.0 issues JESSPOOL checks to verify access to a given output file.

Users can only view STC-type output of started tasks which they are authorized to start. SYSLOG information, which can be viewed through the ATTACH JOB SYSLOG command, is STC-type output.

In the following example, the PROCNAME SYSLOG is added to the STC Record and then the ACID is PERMITted to the ALL Record for global use.

```
TSS ADDTO(STC) PROCNAME(SYSLOG)
               ACID(XYZ)
```

```
TSS PERMIT(ALL) ACID(XYZ)
```

Users can only view TSU-type output that has a jobname that matches an ACID for which they are authorized to submit.

CA Roscoe Batch Utilities Protection

When the CA Top Secret TSSRXBEX is linked with the CA Roscoe batch utilities (ROSCOPY, ROSDATA, LIBSERVE, and LIBUTIL), users can only access and/or modify library members of other CA Roscoe users whose ACIDs are permitted for job submission; see Combining the Interface With the BEXEXIT Module, for detailed instructions on linking BEXEXIT.

The JCL to link-edit the BEXEXIT with the CA Roscoe batch utilities ROSCOPY, ROSDATA, LIBSERVE, or LIBUTIL can be found on the distribution tape in the file named TSSROSCO in the member INSTALL1.

Administration Monitor

The TSS command can be used under CA-Roscoe to perform all security administration. The syntax of the command is identical to that of all other facilities. All functions of the TSS command are supported.

Once installed, the CA Top Secret monitor can be used in several ways:

- Typing TSS followed by a command, executes the command and directs output back to the terminal. (This is the most commonly used mode of the CA Top Secret monitor.)
- Typing TSS or TSS-X repeatedly prompts the user for TSS commands. A user can terminate this prompting mode by entering END.
- Typing TSS-I instructs the monitor to retrieve commands from the AWS and directs output to the terminal.

  Note: When continuing a TSS command to the next line, you must end the line with a space and a dash (-) before beginning the next line.
- Typing TSS-O prompts the user for commands and directs output to the AWS.
- Typing TSS-O followed by a command, executes the command and directs output to the AWS.

Simulation Monitor

All functions of the CA Top Secret Security Simulation Utility are available to CA-Roscoe users through the SIM monitor command. (See the *Troubleshooting Guide* for TSSSIM command functions.) SIM-I and SIM-O perform functions similar to the TSS monitor commands.

Customizing the Interface

This section includes some customizing options that you can tailor to your environment. Included are such features as creating an CA-Roscoe test facility in the Facilities Matrix, and combining the CA-Roscoe/ CA Top Secret interface with your CMDEXIT and SUBEXIT modules.

Creating a CA Roscoe Test Facility

As delivered, the Facilities Matrix does not come with a region for CA Roscoe testing. To add this new region, you must change the CA Top Secret PARM library using the FACILITY control option as follows:

```
FACILITY(USER1=NAME=ROSTEST)
FACILITY(ROSTEST=PGM=ROS)
FACILITY(ROSTEST=MODE=FAIL,LOG(INIT,SMF,MSG),KEY=8)
FACILITY(ROSTEST=UIDACID=8,LOCKTIME=0,DEFACID(*NONE*))
FACILITY(ROSTEST=ACTIVE,NOABEND,ASUBM,NOAUDIT,AUTHINIT)
FACILITY(ROSTEST=INSTDATA,LCFCMD,LUMSG)
FACILITY(ROSTEST=MULTIUSER,NOPROMPT,RNDPW,RES)
FACILITY(ROSTEST=SIGN(M),SHRPRF,NOTRACE,STMSG)
FACILITY(ROSTEST=NOTSOC,WARNPW,NOXDEF)
FACILITY(ROSTEST=NODORMPW,NONPWR)
```

Note: USER1 can be replaced by any available entry in the Facilities Matrix.

Do *not* alter the following default attributes:

```
NOABEND   ASUBM   AUTHINIT   MULTIUSER   NONPWR
NOPROMPT  RES     SHRPRF     NOTSOC
```

Then, create the ACID for the facility:

```
TSS CREATE(ROSTEST) NAME('TEST ROSCOE')
                    FACILITY(STC,BATCH)
                    PASSWORD(NOPW,0)
                    MASTFAC(ROSTEST)
                    DEPARTMENT(dept)
```

Use the following command to define this facility to CA Top Secret as a started task:

```
TSS ADDTO(STC) PROCNAME(ROSTC2)
               ACID(ROSTEST)
```

To define the test facility to CA Top Secret, use the following steps after the TSS CREATE(ROSTEST) command function procedure:

- The ACID associated with the batch job must include the attributes listed in the next step so that ROSTEST is able to bypass further security through NORESCHK once a RACHECK has been issued.
- Define the region ACID using the NORESCHK attribute.
- Allow CA Roscoe access to all required data sets and volumes or add the bypass attributes NODSNCHK and NOVOLCHK to the region ACID.
- Code a USER=ROSTEST keyword on the batch CA-Roscoe job statement.

Combining the Interface With the BEXEXIT Module

If your site wants to change the ACID name from the one derived by CA Top Secret (from the CA-Roscoe key), use the following procedure:

- Write a CSECT that is part of BEXEXIT
- Conventions for the BEXEXIT are:
  - It is *not* reentrant.
  - It must adhere to standard OS linkage conventions.
  - Register 1 contains the address of a two full word parameter list where the first full word contains the address of the CA-Roscoe BEXDSECT and the second full word contains the address of an eight-character field that contains the CA Top Secret derived ACID.
  - Change the ACID name by replacing the area in storage addressed by the second full word. Return with Register 15 equal to zero to continue the authorization check, or not equal to zero to fail the authorization check.
  - Sample source code that can be used with BEXEXIT can be found on the distribution tape in the file named TSSROSCO of the member SAMPBEX.

The JCL to link-edit user-written code with CA Top Secret-supplied code into a load module can be found on the distribution tape in the file named TSSROSCO of the member INSTALL2.

Combining the Interface With the CMDEXIT Module

Note: With CA-Roscoe 6.0, the CMDEXIT is not required *unless* you have defined LCF rules for protecting CA-Roscoe commands and monitors or if you are implementing the CA Top Secret automatic terminal lock facility.

The CA Top Secret command interface, CMDEXIT, must be called if command limiting (LCF) or terminal locking is to be used. The user-written code is front-ended to the CA Top Secret-supplied code. The name of the CSECT is your choice, and the code must be reentrant.

- To incorporate the CA Top Secret interface, place the following instructions in your CMDEXIT module at the required locations:
- At the start of your module, you must save the incoming parameter list address:

```
ST     R1,$PLIST
```

- At the end of your module, call the CA Top Secret interface:

```
LR     R2,R15           SAVE YOUR RETURN CODE
LA     R13,savearea     18 FULL WORD SAVE AREA
L      R1,›PLIST        RESTORE P-LIST ADDRESS
CALL   TSSRXCMD         INVOKE CA Top Secret
CH     R15,=H'8'        TSS SAYS FAIL THE REQUEST?
BNL    *+6              BR YES.USE TSS RETURN CODE
LR     R15,R2           RESTORE YOUR RETURN CODE
```

Sample source code that can be used with your CMDEXIT can also be found on the distribution tape in the file named TSSROSCO of the member SAMPCMD.

The JCL to link-edit user-written code with CA Top Secret-supplied code into a load module can be found on the distribution tape in the file named TSSROSCO of the member INSTALL2.

Combining the Interface With the SUBEXIT Module

The CA Top Secret submit interface, SUBEXIT, is called to perform job submit validation to determine whether the ACID of the signed on user is authorized to submit the ACID associated with a particular job. The user-written code is a front-end to the CA Top Secret-supplied code. The name of the CSECT is your choice, and the code must be reentrant.

To incorporate the interface, place the following instructions in your SUBEXIT module at the required locations:

- At the start of your module, you must save the incoming parameter list address:
  ```
  STR1,$PLIST
  ```
- At the end of your module, call the CA Top Secret interface:
  ```
  LR      R2,R15         SAVE YOUR RETURN CODE
  LA      R13,savearea   18 FULL WORD SAVE AREA
  L       R1,$PLIST      RESTORE P-LIST ADDRESS
  CALL    TSSRXSUB       INVOKE CA Top Secret
  CH      R15,=H'12'     TSS SAYS FAIL THE SUBMIT?
  BNL     *+6            BR YES...USE TSS RETURN CODE
  LR      R15,R2         RESTORE YOUR RETURN CODE
  ```

In CA-Roscoe UXDSECT, there are four fullwords (UXWKAREA) reserved for installation use. The CA Top Secret CSECT, TSSRXSUB, uses the first fullword. Therefore, sites linking in their own SUBEXIT with the CA Top Secret CSECT *must not use the first fullword.* The second through fourth fullwords are available for site use.

The CA Top Secret SUBEXIT supports checking records that replace the record passed to the user-written SUBEXIT (entry code 4, exit code 8). If UXSUBRCO is not a zero on entry to the CA Top Secret SUBEXIT, all replacement records are checked since CA Top Secret assumes you are trying to replace records. It is, therefore, advisable to clear this field after use.

Sample source code that can be used with your SUBEXIT can also be found on the distribution tape in the file named TSSROSCO of the member SAMPSUB.

The JCL to link-edit user-written code with CA Top Secret-supplied code into a load module can be found on the distribution tape in the file named TSSROSCO of the member INSTALL2.

Interfacing With the DSAEXIT Module

CA-Roscoe calls the DSAEXIT to protect the internal reader. The default action of the CA Top Secret DSAEXIT is to allow access to the internal reader.

If your site wants to protect the internal reader, you must write a program having the CSECT name TSSUXDSA. Include this CSECT in the DSAEXIT so that it replaces the TSSUXDSA CSECT in the TSSRXDSA load module. Follow the coding conventions documented in the *CA-Roscoe Extended Facilities For System Programmers*.

The program should be written as if it receives control from CA-Roscoe and returns control to CA-Roscoe. The installation program receives control only for DSAEXIT entry code 30 (protect internal reader).

Administration Menus Installation

Administration Menus simplify the administrator's task of managing users and resources and assigning authorizations. Menus provide a "fill-in-the-blank" approach to administering TSS commands under CA-Roscoe/ETSO.

1. The CA Top Secret Administration panels are written in C language and do not run as an ISPF Dialog Manager Application. The panels use the CA C-Runtime r 3.0 library, which is a part of the CA Common Services for z/OS.

To use the CA Top Secret Administration Panels in a CA-Roscoe/ETSO environment:

1. Assure that the following CA Common Services for z/OS are included in the ROSCOE startup procedure ETSOLIB concatenation, and that both libraries are APF-authorized:
   - CA Top Secret CAILIB
   - CA-Common Services CAILIB (with the CA-C Component installed)
2. Add TSSRPTSS and TSSRPTSM to the CA-Roscoe EPL as shown below:
   - Sign on to a ROSCOE key with library privileges (either through security or through UPS).
   - Issue the command: f RO.ETSOPGMS (If your EPL is stored in a different member, attach that member.)
   - Issue the command: a
   - Insert the following lines alphabetically into the list that is displayed:

     ```
     TSSRPTSM 010 1000 000512 000032 005000 003000 N Y CP TSS ADMIN MODI
     TSSRPTSS 010 1000 000512 000032 005000 003000 N Y CP TSS ADMIN CMDS
     ```
   - Issue the command: SET PRIV ON.
   - Issue the command: u *
3. To access the menus for TSS MODIFY commands, authorized users can issue the command: call tssrptsm
4. To access the menus for all other TSS commands, authorized users can issue the command: call tssrptss

Chapter 9: Implementing Security for DB2

This section contains the following topics:

DB2 Considerations

The DB2 sets are never directly accessed by the user. They are accessed by the DB2 region, and in some cases directly by a DB2 utility. A user never logs on to a DB2 system, the user accesses the DB2 resources through a connection. Therefore, facility controls, such as MODE, for DB2 resources apply to the facility that the user is logged on to, such as TSO.

To simplify the planning and implementation for DB2, brake the DB2 resources down into:

System Objects

These are the resources used by DB2 to implement the management of the user's data and access to it. System objects include the DB2 Catalog, DB2 Directory, Archive Logs, Bootstrap Data Set, and Communications Database. These resources are implemented by DB2 region and certain DB2 utilities. Program protection can be implemented to control which users are allowed access to the utility programs and STC, and what procedures can be used to bring up the DB2 regions.

Data Objects

These objects are how the user's data is organized and accessed by the user. These include databases, tables, plans, system privileges. A separate resource class exists for each of these data objects. The DB2PLAN resource class is the only DB2 resource class that has default protection. Turn on default protection for several other of these resource classes especially the DB2SYS resource class.

DB2 Implementation Strategy

Implement DB2 security in stages:

- Focus on the production subsystems and gradually work towards the test subsystems. Start with subsystem protection. Define who can access each DB2 subsystem, and then turn on protection for that subsystem.
- Permit access to the data objects and selectively turn on default protection for the DB2-related resource classes.
- Define protection and permit access to the system objects.
- Follow the same mode strategy defined for other facilities. However, the mode setting used is that of the mode of the facility associated with the DB2 subsystem.

As soon as the CA Top Secret base product is installed, you can start defining DB2 resources and permitting access to them. CA Top Secret for DB2 does not have to be installed until you are ready to activate your definitions.

DB2 Subsystems

The two main considerations regarding DB2 subsystems are which subsystems should be protected and who should have access to them. When CA Top Secret for DB2 is installed native DB2 security is disabled.

CA Top Secret for DB2

The control of DB2 resources is accomplished using standard CA Top Secret methods. All new DB2 resources have full scope checking and administrative authority support which eliminates the need for secondary authorization IDs and the cascading revoke problems. The direct benefits of CA Top Secret for DB2 are:

- The DB2 resources are easily administered with the same TSS command or the administration panels.
- In CA Top Secret for DB2, the concept of ownership through the creation of an object is eliminated. Instead, all of the DB2-related resources are preferably owned by a department and their use is authorized to users with appropriate privileges.
- With CA Top Secret for DB2 you do not need secondary authorization IDs. They obscure lines of individual accountability.
- Support and security exist for all categories of DB2 privileges and authorities. Because the SYSADM authority has complete control over most DB2 resources, you should carefully limit and monitor its use as you would an MSCA.
- There are discrete checks with unique class names identifying the type of function secured.
- Specific class names permit matching of relationships with existing DB2 controls.
- Access levels are supported as applicable to each function.
- All auditing and violation activity within DB2 is recorded to SMF and/or the Audit/Tracking File. All current facilities for reporting, including the online TSSTRACK reporting utility, are supported.
- The Catalog Synchronization Utility provides the ability to bring DB2 catalog entries up-to-date with CA Top Secret for DB2.

Chapter 10: Implementing Security for Other CA Products

This section contains the following topics:

CA 1

This interface allows CA Top Secret to support full 44-character data set name definitions and protection of tapes. In addition, many CA 1 unique resources and functions can be protected using CA Top Secret. CA 1 determines what is protected by using CA 1 parameter file options.

All resources used by CA 1 that can be protected by CA Top Secret are listed below. The list also includes a brief description of what security they provide.

DATASET(dsn)

Provides full 44-character data set name protection; administered like disk data sets.

CACMD(L0cmd)

Controls commands for online users. The resource name is set to L0 plus the first six characters of the command name. Access to these CA 1 commands can be controlled: CLEAN, EXTEND, EXPIRE, RETAIN, DELETE, ADD, CHECKIN, and CHECKOUT.

CATAPE(YSVCCOND/YSVCUNCD)

Controls access to the CA-1 Y SVC, which controls all direct access to the TMC and audit data sets.

CATAPE(NLRES/NLNORES/NSLRES/NSLNORES/BLPRES/BLPNORES)

Controls which users are permitted to NL, NSL, and BLP processing for tapes (whether they are under CA 1 control).

CATAPE(FORRES/FORNORES)

Controls 98000 (foreign) tape processing. These resources determine which users can read a foreign tape or write to a CA 1 tape having 98000 specified.

CATAPE(password)

Controls the use of online user passwords that determine which internal fields can and cannot be updated.

In addition to securing the desired CA 1 resources, you must also:

- Set the TAPE control option to OFF
- Ensure that the CA 1 initialization STC (TMSINIT) has update access to YSVCCOND and YSVCUNCD

CA 7

CA-7 external security includes security for signon and signoff, top line commands, panel access and functions within a panel, job submission authority, data set security, and job owner ID security.

An explanation of how to implement the interface appears in the CA 7 documentation set. In addition to these instructions, the CA Top Secret security administrator must create a region control acid and started task definition for both the CA 7 online STC (CA7ONL) and the CA 7 Independent Communications Manager (CA7ICOM).

Example: region control and started task definitions

```
TSS CREATE(CA7ONL) NAME('CA-7 ONLINE ACID')
                   FACILITY(STC,BATCH)
                   TYPE(USER)
                   PASSWORD(NOPW,0)
                   DEPARTMENT(CA7OPS)
                   MASTFAC(CA7)
                   NOSUBCHK

TSS ADDTO(STC) PROCNAME(CA7ONL)
               ACID(CA7ONL)

TSS CREATE(CA7ICOM) NAME('CA-7 ICOM')
                    FACILITY(STC)
                    TYPE(USER)
                    PASSWORD(NOPW,0)
                    DEPARTMENT(CA7OPS)
                    MASTFAC(CA7)
                    NOSUBCHK

TSS ADDTO(STC) PROCNAME(CA7ICOM)
               ACID(CA7ICOM)
```

CA Top Secret comes with CA 7 defined as a default facility in the Facilities Matrix. The following suboptions are required:

```
TSS MODIFY(FACILITY(CA7=ASUBM,MULTIUSER,NOABEND,PGM=SAS,LOG(ALL)))
```

To allow a user to access the CA 7 online region, enter:

```
TSS  ADDTO(USER)  FACILITY(CA7)
```

All CA 7 command security is administered using the PANEL resource class.

Note: If CA Top Secret is *not* used to secure COMMANDs under CA 7, ACIDs having access to FACILITY(CA7) must also be defined in the CA 7 SECURITY GEN.

Implement Security for CA Datacom

CA Datacom provides external security for all product resources, functions, and accessors. In addition to these instructions, the CA Top Secret security administrator must also create a facility, a region control ACID, and a started task definition for each CA Datacom Multi-User Facility (MUF).

To create each CA Datacom MUF facility, enter the following commands:

```
TSS MODIFY(FACILITY(USERnn=NAME=PRODMUFx)
```

```
TSS MODIFY(FACILITY(PRODMUFx=MULTIUSER,PGM=***,NORES,SIGN(M),AUTHINIT)
```

```
TSS MODIFY(FACILITY(PRODMUFx=SHRPRF,NOABEND)
```

To create a region control ACID and add the started task to the STC table, enter the following commands:

```
TSS CREATE(MUFxACID) NAME('PRODMUFx CNTL ACID') TYPE(USER)
                     DEPARTMENT(dept)
                     FACILITY(STC,BATCH)
                     MASTFAC(PRODMUFx)
                     PASSWORD(NOPW,0)
```

```
TSS ADDTO(STC) PROCNAME(MUFxSTC)
               ACID(MUFxACID)
```

To allow a user to access the CA Datacom MUF, enter the following command:

```
TSS  ADDTO(USER)  FACILITY(PRODMUFx)
```

CA Datacom products share the following CA Datacom resources, which are protected by CA Top Secret. Also included in this list is a brief description of what security they provide.

Note: For complete information about resources and using external security for CA Datacom, see the *CA Datacom Security Reference Guide*

DTSYSTEM(*cxxname.product*)

Secures system product combinations. A system includes all of the databases and tables defined in the CA Datacom directories (C*xx*). The product is identified by a two-character product code:

DB

CA Datacom/DB

DD

CA Dataquery CA Datacom Datadictionary

DQ

CA Dataquery

DTADMIN(*cxxname.product*)

Secures product administrator authority. A user who has access to the DTADMIN(*cxxname*.DB) is considered a Global Owner and can create a schema for SQL, issue GRANT or REVOKE for any SQL controlled tables, and drop any table.

DTTABLE(*cxxname*.DB00*nnn.table*)

Determines which users can access CA Datacom/DB tables. The resource is made up of the C*xx* name, followed by the database and table name.

DTUTIL(*cxxname*.DBUTLTY.*function.subfunction*)

Protects the desired CA Datacom DBUTLTY utility functions and serves as an alternative method for console operators to secure unprotected CA Datacom console commands.

DTUTIL(*cxxname*.DB0*nnnn.table.right*)

Protects CA Datacom DBUTLTY utility functions that include table access.

DTUTIL(*cxxname*.DQ*utility.function*)

Protects processes in CA Dataquery.

DTUTIL(*cxxname*.DD0*nnnn*.DD*utility.function*)

Protects processes and utilities in CA Datacom Datadictionary.

DTUTIL(*cxxname*.DD00*nnn.table.status.function*)

Protects entity-level security in CA Datacom Datadictionary.

Chapter 11: Implementing Security for Non-CA Products

This section contains the following topics:

ADABAS

The ADABAS External Security Interface (ESI) allows a standard security interface between ADABAS and CA Top Secret. ADABAS ESI has been provided by Software AG since ADABAS 5.2.

Activating the CA Top Secret ADABAS Interface

The CA Top Secret ADABAS interface operates through a dynamic intercept of the ADABAS SVC. If the ADABAS control option is present and has the correct SVC numbers, the installation is automatic. The correct control option setting for ADABAS is:

```
ADABAS(nnn)
```

nnn

The SVC number (decimal) of the SVC generated for use by ADABAS.

Resource Control

Use the CA Top Secret ADABAS interface to supplement the security controls provided by the basic ADABAS system to protect ADABAS databases and files. CA Top Secret does not eliminate or supersede any of the controls and logical relationships provided by the ADABAS data dictionary.

This interface introduces the resource DATABASE. DATABASE is an owned resource, with WHOHAS and WHOOWNS support, scope checking, and all the administration controls implied for an ownable resource.

The syntax for ADDing and PERMITting DATABASE databases and files is:

```
TSS ADDTO(acid) DATABASE(DdddFfff)
TSS PERMIT(acid) DATABASE(DdddFfff) options...
```

ddd

The database number from 1 through 255.

fff

The file number from 1 through 255.

Two access levels are available for ADABAS files:

READ

Read-only access can be performed on the database/file combination.

UPDATE

Read, write, or update operations can be performed on the database/file combination.

For example, to protect file number 105 (contained within the ADABAS database number 3) from unauthorized access, the administrator assigns ownership to the Payroll Department, enter:

```
TSS ADDTO(PAYROLL) DATABASE(D003F105)
```

To then allow all users associated with the payroll profile, PAYPROF, to update this file, enter:

```
TSS PERMIT(PAYPROF) DATABASE(D003F105) ACCESS(UPDATE)
```

There are certain ADABAS commands that can only be used by ADABAS utilities. These commands are protected by the OTRAN resource. These two character commands are: A9, SP, L7, L8, LA, LB, LC, and LD.

To protect all of these commands, use this TSS command function:

```
TSS ADDTO(DEPT01)  OTRAN(ADABAS)
```

To permit use of any of these commands, enter:

```
TSS PERMIT(USER01) OTRAN(ADABASxx)
```

xx

Represents any of the two character commands listed above.

Modes of Operation

Because ADABAS is processed as an access method rather than as a facility, there is no unique mode assignment to ADABAS. Instead, the MODE for the user who accesses a resource is determined by the MODE selected for that user in the facility (CICS, TSO, or BATCH) used to access ADABAS. The usual CA Top Secret convention of modes overriding in the order of user, profile, facility, and global, applies. In addition, ACTION(FAIL) is honored for processing specific PERMITs in FAIL mode.

Signon Controls

Signon controls are not directly applicable to the ADABAS interface. Signon is authenticated by the facility used to access the ADABAS databases and files. Specifically, for the purposes of this interface, signons to CICS, TSO, or BATCH are controlled by conventional means.

Logging

Logging for the DATABASE resource is controlled by the LOG control option in effect for the facility through which the ADABAS databases and files are being accessed. The following options are available:

ACTIVITY

Logs all activity in the host facility.

INIT

Logs all signon and signoff activity (job initiations for BATCH).

ACCESS

Logs all resource access in the host facility.

ALL

Selects all log options to be active.

NONE

Deactivates logging in the host facility, except for violations and audited events.

MSG

Requests writing of violation messages to the user's terminal (or job log for BATCH).

SMF

Requests writing of log data to SMF.

SEC9

Forces violation messages to be displayed on the MVS security console (ROUTCDE9).

COMPLETE

CA Top Secret comes with COMPLETE (a product of Software AG of North America, Inc.) already defined as a default facility in the Facilities Matrix. This means that the security attributes that control CA Top Secret processing for COMPLETE are predefined. These attributes are actually suboptions of the FACILITY control option.

Note: COMPLETE is defined to CA Top Secret, through the Facilities Matrix, as a multi-user address space.

The COMPLETE default options are:

```
INITPGM=THR      id=C      TYPE=21
ATTRIBUTES=ACTIVE,SHRPRF,ASUBM,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,NORNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFCMD
ATTRIBUTES=NOTRACE,NODORMPW,NONPWR,MSGLC
MODE=FAIL LOGGING=INIT,MFG,SEC9
UIDACID=8 LOCKTIME=000 DEFACID=*NONE* KEY=8
```

If the default attributes are not applicable to your site, they can be changed in the Parameter File using the FACILITY control option.

DB2

The following modifications must be made if you are using CA Top Secret to secure DB2 resources (DB2 is a product of the IBM Corporation).

- Define DB2 started tasks ssssMSTR, ssssDIST and ssssDBM1 (where ssss is the four-character subsystem ID) to CA Top Secret through a TSS ADDTO(STC).

  Also add the IMS Resource Lock Manager (IRLM) as a started task. DB2 uses the IRLM to manage the locking of DB2 resources. The name for the IRLM started task was specified during the DB2 install process.

- Use the DB2 resource keyword for connection authorization in the format (where *ssss* is the subsystem name your site is using for DB2):

  ```
  DB2 \{(DSNR.ssss.BAT)\} for BATCH and TSO connections
  DB2 \{(DSNR.ssss.DIS)\} for Distributed Data Facility (DDF)
  DB2 \{(DSNR.ssss.MAS)\} for connection from IMS
  DB2 \{(DSNR.ssss..SAS)\} for connection from CICS
  DB2 \{(DSNR.ssss.RRSAF)\} for connection from RSS Attachment Facility
  ```

  Note: CA Top Secret r4.4 and above supports DB2 resources previously defined as ABSTRACTs. However, it is recommended that the DB2 resource class be used for newly defined DB2 resources.

  For connection authorization, individual user checking is not performed for CICS and IMS; the ACID associated with the CICS and IMS started tasks or the associated region ACID is checked instead. The ACID associated with a job is used to validate the connection for TSO and BATCH.

- To protect archive log tape data sets, specify YES on the IBM DB2 installation CLIST panel entry for ARCHIVE LOG RACF.

PERMIT users with appropriate access levels to required DB2 (O/S) data sets.

DXT Utility

Data to be extracted by the DXT utility is protected using a pseudo-data set name generated in the format:

```
DXT.dddd
```

dddd

A one to 26 character data element (file, PCB, or view).

After adding DXT(DXT.) to the appropriate department, individual, or profile, ACIDs can be permitted access to required data.

DB2 Support

CA Top Secret r 4.4 and above supports DB2 Version 2 SQL for Secondary Authorization IDs. Secondary Authorization IDs are defined to CA Top Secret through the IBMGROUP resource. At logon, CA Top Secret automatically builds a list of authorized IBMGROUPs for use by DB2. Since this list is built at logon, access qualifiers such as time of day and day of week are only validated at that time. The list of IBMGROUPs is not refreshed until the next logon.

Note: The CA Top Secret DB2 exits provided in r 4.2 of CA Top Secret are no longer necessary. The DB2 authorization exits (DSN3@ATH and DSN3@SGN) provided by IBM, however, are.

Connecting to DB2 With CICS

In a CICS environment, the value of the primary AUTHID is dependent on how the AUTH= parameter is coded in the CICS RCT. There are several valid parameters for AUTH=; the most important ones are described below.

AUTH=GROUP

Uses the signon CICS userid as the primary AUTHID. A signon is not driven for this event. This performs the same function as AUTH=USERID with the added benefit of improved performance.

AUTH=USERID

Uses the signon CICS userid as the primary AUTHID. A signon is driven for this event even though the userid is already signed on to CICS.

AUTH='acidname'

Uses the acidname as the primary AUTHID. A signon is driven if the acidname is a valid CA Top Secret ACID with access to the CICS facility being used.

AUTH=TXID

Uses the CICS transaction name as the primary AUTHID. A signon is driven if the primary AUTHID and the transaction name are valid CA Top Secret ACIDs with access to the CICS facility being used.

CA Top Secret IMS ACEE Locater Subroutine

The DB2 DSN3@SGN exit looks at the ACEE to determine the groups that are used as the DB2 Secondary Authorization IDs. In an IMS environment, if the ACEE is not available, the exit issues a RACROUTE REQUEST=VERIFY,ENVIR=CREATE call to obtain the ACEE. These RACROUTE calls can produce a great deal of overhead.

CA Top Secret /MVS provides a subroutine, TSS3@LOC, to locate the ACEE of an IMS user. By using this subroutine, called by the DSN3@SGN exit, you can locate the ACEE of an IMS user without resorting to a RACROUTE call.

To use the subroutine, the following conditions must exist:

- You must modify RACF SAMPLE version of the DSN3@SGN exit to call this subroutine.

  Note: The subroutine can be called in all environments by the DB2 DSN3@SGN exit, but only returns the ACEE in an IMS MPP or BMP environment.

- CA Top Secret MVS must be running in the IMS environment, with ISIS=1 specified in the control region.

The following entry conditions apply:

- TSS3@LOC must be entered with standard linkage conventions, namely:
  - R13
  - Address of an available save area
  - R14
  - Return address
  - R15
  - Entry address
- The subroutine is entered in the same state and mode as the DB2 DSN3@SGN exit, currently:
  - Supervisor state
  - Key 7
  - AMODE 31
- At entry, R1 must point to a parameter list. The format of the parameter list is:
  - Address of DB2 EXPL control block
  - Address of DB2 AIDL control block
  - Address of work area (optional)
  - Length of work area passed (optional)

  If the optional parameters are not included or if 0 is not specified, the program performs a GETMAIN/FREEMAIN for its work area.

The following exit conditions apply:

R15

Return code from the subroutine

0

ACEE found and returned in R1

4

ACEE not found; caller must issue RACROUTE REQUEST=VERIFY,ENVIR=CREATE

8

Unsupported environment (not MPP/BMP with CA Top Secret)

16

Subroutine encountered logic error

R1

Address of ACEE if R15=0. If R15 does not equal 0, R1 will contain a 0.

DDF Support

If you are using the DB2 2.3 DDF function, you should associate the address space DB2DIST with a DB2 facility using a MASTFAC. The users must be granted access to the facility with the appropriate connection resources- DSNR.subsys.DIST. Your steps to properly set up the DB2 facility are:

- Define the DB2DIST started procedure to the STC record and associate it with an ACID
- Define a facility for DB2/DDF
- Give the DB2DIST ACID a MASTFAC of the defined facility

A sample facility definition is provided below:

```
INITPGM=***    id=33  TYPE=100
```

```
ATTRIBUTES=IN-USE,ACTIVE,SHRPRF,ASUBM,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,RNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFTRANS
ATTRIBUTES=MSGLC,NOTRACE,NOEODINIT,IJU,NODORMPW,NONPWR
MODE=WARN DOWN=GLOBAL LOGGING=INIT,SMF,MSG,SEC9
UIDACID=8  LOCKTIME=000  DEFACID=*NONE*  KEY=8
MAXUSER=03000 PRFT=003
MODIFY  FUNCTION  SUCCESSFUL
```

Note: Remember to specify TYPE=DB2 in the Facility Matrix Table to display the correct type of 100.

FOCUS

FOCUS is a product of Information Builders, Inc. The FOCUS/CA Top Secret security interface supports the concept of an inferred FOCUS password-ID. When a user signs on to FOCUS, he usually needs to identify himself with one of the following:

- SET USER=password
- PASS password
- -PASS password (dialog manager command)

With the interface activated, the signon sequence can be automated- thus eliminating the user's need to remember another password-ID.

Installation

The interface is distributed in object-module format. The installation procedure is straightforward and consists of two link-edit steps, outlined as follows:

- Link-edit the FOCUS Security Interface. This interface must be present before the second link-edit is performed. JCL for this step should be located in your FOCUS JCL library, member name JCLACF21.
- Link-edit the CA Top Secret Security Interface. The CA Top Secret interface uses the FOCUS signon security interface, but replaces the CSECT named FOCUSID with an CA Top Secret version. This gives CA Top Secret the ability to interrogate the user's installation data (INSTDATA) field for the appropriate FOCUS password.

  Sample JCL for this is shown next:

```
//LKED     EXEC PGM=IEWL,PARM='<<standard FOCUS lked parms>>'
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD        DSN=&&SYSUT1.,UNIT=SYSDA,SPACE=(1024,(50,20))

//FOCCNTL  DD        DSN=Name.of.FOCUS.lked.cntl.statements,DISP=SHR
//FOCLIB   DD        DSN=Name.of.FOCUS.loadlib,DISP=SHR
//TSS      DD        DSN=Name.of.TSSOPMAT,DISP=SHR
//SYSLMOD  DD        DSN=Name.of.FOCUS.loadlib,DISP=SHR
//SYSIN    DD  *
INCLUDE  TSS(FOCUSID)    TSS interface
INCLUDE  FOCCNTL(FOCUS)  FOCUS overlay structure lked cntl
INCLUDE  FOCLIB(FOCUS)   Basic FOCUS module w/security
NAME     FOCUS(R)
```

Note: Beginning with r6, FOCUSID is no longer supplied by default in the FOCUS product. This means that FOCUSID must be link edited according to the instructions shown with the JCL above.

Inferred Passwords

By placing the user's password-ID in the installation data field (INSTDATA) of the user's Security Record, CA Top Secret returns the password-ID to FOCUS when FOCUS is invoked by the user. This password-ID is the inferred password.

In addition, several CA Top Secret users can be mapped into the same FOCUS password-ID. This is quite useful if the password-ID represents a group of users who have identical FOCUS authorization privileges.

The INSTDATA Field

Since the INSTDATA field can be used for multiple purposes by various users, a simple syntax scheme for INSTDATA must be used to distinguish different parameters. The syntax rules are:

- Parameter items can be simple attributes (one- through eight- character alphanumeric symbols) or keywords (one- through eight- character alphanumeric symbols followed by a value enclosed in parentheses).
- Parameter items must be separated by commas.
- Values enclosed in parentheses for keyword parameters can be any string of characters. If the character string contains any blanks or special characters, then the entire string must be delimited by single quotation marks.
- The FOCUS password-ID is specified in the installation data field using FOCUS(xyz), where xyz is the user's password-ID. The following TSS command function is used to add the FOCUS password-ID:

  ```
  TSS ADDTO(user) INSTDATA(FOCUS(xyz))
  ```

The following examples indicate multi-use INSTDATA fields.

```
'ATTRIB1,ATTRIB2,KEYWORD1 (VALUE1),FOCUS(MYPW)'
ATTRIB1,FOCUS("MYPW."), ATTRIB2,KEYWORD1(VALUE1,VALUE2)
```

In the first example, note the use of single quotation marks to delineate strings containing blanks or special characters.

In the second example, note the use of the FOCUS password-ID ending in a period. Under normal circumstances, an inferred FOCUS password can be overridden by a user after FOCUS is invoked by entering one of the explicit password identification commands. This redefinition lasts for the duration of the FOCUS session, or until the user redefines another FOCUS password-ID. In certain circumstances, this inferred password override is desirable; however, in most cases it is not. To prevent users from redefining a new FOCUS password-ID after invoking FOCUS, simply end the password-ID with a period. The period is stripped away before comparing the password with those defined in the FOCUS masterfile.

Note: The period has special significance only during initial inferred password processing.

Security Administration Considerations

Since this interface provides support for an inferred FOCUS password-ID and does not replace existing internal FOCUS security (but rather automates the user's signon process to FOCUS), the CA Top Secret security administrator must:

- Place the user's FOCUS password-ID in the INSTDATA field, and
- Allow the user to access various data sets only with the FOCUS program (perhaps on a restricted basis).

The ADD command function is used to place the FOCUS password-ID into the user's Security Record. The INSTDATA keyword for the FOCUS password-ID is FOCUS. The following examples illustrate the use of the TSS command for TSS/FOCUS password-ID administration:

In the following example, the user, SYS001, has a FOCUS password-ID of SYSX and is allowed to redefine his FOCUS password (for the duration of the FOCUS session) after invoking FOCUS.

```
TSS ADDTO(SYS001) INSTDATA(FOCUS(SYSX))
```

In this example, the user, PAY100, has a FOCUS password-ID of PAYX and is not allowed to alter his inferred FOCUS password-ID after invoking FOCUS.

```
TSS ADDTO(PAY100) INSTDATA(FOCUS("PAYX."))
```

The PERMIT function of the TSS command is used to handle the second portion of CA Top Secret /FOCUS security. In the following examples, assume that the FOCUS program at your site has been installed under the name FOCUS.

The first example shows FOCUS accessing payroll production data sets beginning with the high-level qualifier PAYROLL. In addition, ten junior-level employees in the PAYROLL Department, using profile PAY001JR, are only allowed access to these data sets using FOCUS.

```
TSS PERMIT(PAY001JR) DSNAME(PAYROLL.)
                     PRIVPGM(FOCUS)
                     ACCESS(UPDATE)
```

In the second example, a specific user is accessing accounting data sets beginning with the high-level qualifier ACCTG. on a read-only basis, using FOCUS in addition to several other general-purpose programs.

```
TSS PERMIT(USR001) DSNAME(ACCTG.)
                   ACCESS(READ)
```

HSM

To secure HSM through CA Top Secret:

- Review the Facilities Matrix entry for HSM
- Create a region ACID for HSM
- Add that region ACID to the STC Record
- Ensure that tape security (EI or E) is specified in the HSM startup options

Facilities Matrix

The Facilities Matrix entry for HSM is predefined with the following defaults:

```
INITPGM=ARC id=H
ATTRIBUTES=IN-USE,ACTIVE,SHRPRF,NOASUBM,ABEND,SUAS,NOXDEF
ATTRIBUTES=NOLUMSG,NOSTMSG,SIGN(M),INSTDATA,NORNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,NOWARNPW,NOTSOC,LCFCMD
ATTRIBUTES=NOTRACE,NODORMPW,NONPWR
MODE=WARN   LOGGING=INIT,SMF,MSG,SEC9
UIDACID=8  LOCKTIME=000  DEFACID=*NONE*  KEY=8
```

Consider creating a default ACID for undefined users who might need to access HSM.

Create the HSM Region ACID

Since HSM is treated as a started task, you must create a region ACID with a MASTFAC of HSM and the appropriate bypass attributes. For example:

```
TSS CREATE(ARCHIVE) NAME('HSM REG ACID')
                    TYPE(USER) DEPARTMENT(STOR01)
                    PASSWORD(xxxxxx,0)
                    FACILITY(STC)
                    MASTFAC(HSM)
                    NODSNCHK, NORESCHK, NOVOLCHK, NOLCFCHK
```

Note: OPTIONS(4) should be set in the TSS parameter file.

Add HSM to the STC Record

Once the HSM facility has been defined to the Facilities Matrix and a region ACID has been created, the last step is to add the HSM procname to the STC Record. Continuing from the previous example, enter:

```
TSS ADDTO(STC) PROCNAME(HSM)
               ACID(ARCHIVE)
```

TAPESECURITY HSM Start Option

The TAPESECURITY start option in DFHSM is used to determine how tape management is handled for DFHSM tapes. CA Top Secret is not a tape management package and does not support RACF or RACFINCLUDE under the TAPESECURITY option. The *DFHSM Systems Programmer Guide*, describes security tape processing under DFHSM and the EXPIRATIONINCLUDE and EXPIRATION parameters.

ICSF Support

The Key Store Policy feature adds various XFACILIT class resource profiles that are obtained using RACROUTE EXTRACT by ICSF. These profiles do not contain extractable fields. The extract only determines if the profile exists. CA Top Secret includes a SIGNAL=YES attribute to resclasses in the RDT. If a resource is added or removed from an ACID in which the resclass has the SIGNAL=YES attribute, CA Top Secret issues the ENF62 signal. When ICSF detects this signal, it uses the results of these extracts.

To determine how ICSF uses the extracts, see the ICSF documentation.

ICSF issues extracts for the following resources to determine settings:

- CSF.CKDS.TOKEN.CHECK.LABEL.WARN
- CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
- CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL
- CSF.CKDS.TOKEN.NODUPLICATES
- CSF.PKDS.TOKEN.CHECK.LABEL.WARN
- CSF.PKDS.TOKEN.CHECK.LABEL.FAIL
- CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL
- CSF.PKDS.TOKEN.NODUPLICATES
- CSF.CSFKEYS.AUTHORITY.LEVELS.FAIL
- CSF.CSFKEYS.AUTHORITY.LEVELS.WARN
- CSF.XCSFKEY.ENABLE.AES
- CSF.XCSFKEY.ENABLE.DES

The administrator can indicate SIGNAL=YES on resclasses using the TSS REPLACE command. For example, add the option by issuing the following command:

```
TSS REP(RDT) RESCLASS(XFACILIT) ATTR(SIGNAL)
```

Enable ICSF

To properly return information for ICSF extracts, you must permit the resources and entities to the ACID that is defined to the ICSF started task.

For example, if you want the started task ACID for ICSF to have access to CSF.XCSFKEY.ENABLE.AES, issue the following command:

```
TSS PERMIT(CSFACID) XFACILIT(CSF.XCSFKEY.ENABLE.AES)
```

DLF Resource Class Setup

ICSF issues extracts for the DLFCLASS. The Data Lookaside Facility (DLF) controls the loading of data sets into ESA hyperspace by selected jobs.

To return extracted information for the DLFCLASS, you must issue the following command:

```
TSS ADD(DLF) DSNAME(dsname) JOBNAME(CSF) RETAIN
```

dsname

Indicates the data set name that you are adding to the DLF.

For details regarding this feature and when to issue this command, see the ICSF documentation.

NCCF

CA Top Secret comes with NCCF, a product of the IBM Corporation, already defined as a default facility in the Facilities Matrix. This means that the security attributes that control CA Top Secret processing for NCCF are predefined. These attributes, listed below, are actually suboptions of the FACILITY control option.

Note: NCCF is defined to CA Top Secret, through the Facilities Matrix, as a multi-user address space.

The NCCF default attributes are shown next:

```
INITPGM=DSI     id=N     TYPE=06
ATTRIBUTES=ACTIVE,SHRPRF,ASUBM,ABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,NORNDPW,NOAUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFCMD
ATTRIBUTES=MSGLC,NOTRACE,NOEODINIT,IJU,NODORMPW,NONPWR
MODE=WARN  DOWN=GLOBAL  LOGGING=ACCESS,INIT,MSG,SEC9
UIDACID=8  LOCKTIME=000 DEFACID=*NONE* KEY=8
MAXUSER=03000 PRFT=003
```

Attributes are explained in the *Control Options Guide*.

If the default attributes are not applicable to your site, they can be changed in the Parameter File using the FACILITY control option.

Required Modifications

The following modifications must be made if you are using CA Top Secret security for NCCF:

- CCF must be generated with the NCCF option VERIFY=RACF
- All NCCF users must be defined to NCCF with dummy passwords

CA Top Secret validates facility access. Since there is no password change facility within NCCF, expired passwords must be changed manually by the security administrator.

Omegamon

Two types of security are available with Omegamon:

- Product-level security, which prevents unauthorized access to Omegamon II
- Command-level security, which provides logon and command validation

Defining Product-level Security Using CA Top Secret

To implement product-level security using CA Top Secret, you must perform the following steps:

- Define task as a facility to CA Top Secret in the Facilities Matrix, where task is the name of the started task (for example, OMIIMVS, OMIICSA, OMIIRCOL, and so on). All examples use OMIIMVS as the started task.

  To define the OMIIMVS task, add the following commands to the CA Top Secret Parameter File:

  ```
  FACILITY (USER3=NAME=OMIIMVS)
  FACILITY (OMIIMVS=MODE=FAIL,ACTIVE,SHRPRF)
  FACILITY (OMIIMVS=PGM=KLV,NOASUBM,NOABEND,NOXDEF)
  FACILITY (OMIIMVS=ID=3,MULTIUSER,RES)
  FACILITY (OMIIMVS=LUMSG,STMSG,WARNPW)
  FACILITY (OMIIMVS=NOINSTDATA,SIGN (M))
  FACILITY (OMIIMVS=NORNDPW,AUTHINIT)
  FACILITY (OMIIMVS=NOPROMPT,NOAUDIT,NOTSOC)
  FACILITY (OMIIMVS=LOG (INIT,SMF,MSG,SEC9))
  ```

  Note: If a MASTFAC is defined for the Omegamon region ACID, it is not necessary to define PGM= on the FACILITY definition.

- Create the master facility ACID for Omegamon. An example appears below.

  ```
  TSS CREATE(OMEGAMON) NAME('acid')
                       DEPARTMENT(OMEGDEPT)
                       MASTFAC(OMIIMVS)
                       TYPE(USER)
                       PASS(NOPW,0)
  ```

  acid is the Omegamon master facility ACID.

- Define the Omegamon address space as a started task in the STC Record using the master facility ACID you created in Step 2. A sample definition appears below.

  ```
  TSS ADDTO(STC) PROCNAME(OMIIMVS)
                 ACID(OMEGAMON)
  ```

- Modify the security definition in the hlq.RM2PARM member KLVINNAM as shown below.

  ```
  DEFAULT DSNAME(hlq.NAM) EXIT=KLVTSNEV RACF NODB
  ```

- Install the exit for security validation. Omegamon provides a sample interface in member hlq.TLVSPENU(KLVTSNEV).

  Assemble and link KLVTSNEV with AC=1 into the hlq.TLVLOAD library; sample JCL is provided by Omegamon.

- Define the Omegamon data sets to be protected by CA Top Secret ; this step is optional.

Define Command-level Security Using CA Top Secret

To implement command-level security:

1. Create a facility entry for OMEGAMON with the following facility options: MULTIUSER, PGM=KOB.

2. Add the OMEGAMON facility to users, as shown below.

   ```
   TSS ADDTO(acid) FACILITY(OMEGAMON)
   ```

   Acid is the user's ACID.

3. Define a resource class to the RDT for Omegamon as shown below.

   ```
   TSS ADDTO(RDT) RESCLASS(KOMCANDL) RESCODE(nn)
   ```

   nn is any hexadecimal code between 01 and 3F that is not currently being used for another user-defined resource.

   Note: The resource class name does not have to be KOMCANDL. However, the name chosen must be consistent with the resource in the command for steps 4 through 7 which follow.

4. Give ownership of the resource class using INITIAL as a prefix.

   ```
   TSS ADDTO(acid) KOMCANDL(INITIAL)
   ```

   acid is the department ACID.

   Omegamon issues resource checks using four different levels of authority: INITIAL0, INITIAL1, INITIAL2, and INITIAL3. These levels are associated with Omegamon commands in the Command Table (see Step 8).

5. PERMIT an Omegamon command level to users as shown below.

   ```
   TSS PERMIT(acid) KOMCANDL(INITIAL0)    For issuing level 0 commands
   TSS PERMIT(acid) KOMCANDL(INITIAL1)    For issuing level 1 commands
   TSS PERMIT(acid) KOMCANDL(INITIAL2)    For issuing level 2 commands
   TSS PERMIT(acid) KOMCANDL(INITIAL3)    For issuing level 3 commands
   TSS PERMIT(acid) KOMCANDL('INITIAL ')  To change security levels acid is the user's
   ACID.
   ```

   The first four authorizations lock the user into one command level and disables the /PWD command (even if the user knows the password). The last PERMIT allows a user to change security levels using the /PWD password if the password is known; the trailing blank is required.

   The commands that a user can issue are the ones defined, through an Omegamon table, to be in his command level. Immediately after the RACINIT, a RACHECK is called to assign the user a command level. This is validated internally by Omegamon.

6. Set up rules for each command to be protected by CA Top Secret:

   ```
   TSS ADDTO(dept) KOMCANDL(PEEK)
   TSS PERMIT(acid) KOMCANDL(PEEK)
   ```

   Each command you protect requires that EXTERNAL=YES be specified in the Omegamon Security Table (see Step 8).

Note: If you are securing a command that begins with a slash (/) or a period (.), you must change the command to begin with a dollar sign ($) instead of a slash (/) and an at sign (@) instead of a period (.).

7. Change the Omegamon RACF exit source as described below.

   - Remove the APPLICATION= parameter on the RACIMDL RACINIT MF=L and RACCMDL RACHECK MF=L macros near the end of the routine.
   - Replace the instructions at the beginning of the interface with:

     ```
     MVI U#CHCLS,X'08'
     ```

     ```
     MVC U#CHCLSD,=CL8'KOMCANDLE'
     ```

   - Put the instructions shown above after this instruction:

     ```
     OI  U#CHAUT1,U@CH1RAC
     ```

   - Link the exit as instructed by Omegamon.

8. Set up the Omegamon Security Table by modifying the control statements in member KOMSUPDI in hlq.ROMDATA as follows:

   - Uncomment the MODULE command statement and enter the name of the exit KOMRACFX on the MODULE statement as shown below.

     ```
     MODULE=KOMRACFX
     ```

   - Indicate which commands are to be validated by CA Top Secret rules by setting EXTERNAL=YES on the COMMAND control statements. Also indicate which level the command is to be checked to by setting LEVEL=n, where n is 0, 1, 2, or 3. A sample appears below.

     ```
     COMMAND=APFU,LEVEL=3,EXERNAL=YES
     ```

   - Commands that are to be validated by Omegamon internal security are defined by setting EXTERNAL=NO. In addition, indicate which level the command is to be checked at by setting LEVEL=n, where n is 1, 2, or 3. A sample appears below.

     ```
     COMMAND=XMLS,LEVEL=1,EXTERNAL=NO
     ```

     Note: The LEVEL associated with the COMMAND control statement corresponds to the suffix associated with the INITIAL resource PERMITted to users.

     The default for EXTERNAL is NO.

   - Modify and submit job KOMSUPD to update the Security Table (the JCL is provided by Omegamon).

OPC/ESA

- Define OPC/A as a facility using CA-7 as a model.

  ```
  PGM=CSY      id=?  TYPE=099
  ATTRIBUTES=ACTIVE,SHRPRF,NOASUBM,NOABEND,MULTIUSER,NOXDEF
  ATTRIBUTES=NOLUMSG,NOSTMSG,SIGN(M),INSTDATA,NORNDPW,AUTHINIT
  ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFCMD
  ATTRIBUTES=MSGLC,NOTRACE,NOEODINIT,IJU,NODORMPW,NONPWR
  MODE=WARN  DOWN=GLOBAL  LOGGING=MSG,SEC9
  UIDACID=8 LOCKTIME=000 DEFACID=*NONE*   KEY=8
  MAXUSER=03000  PRFT=003
  ```

- To define a PROCNAME to the STC Table and an OPC/A region ACID, enter the following command:

  ```
  TSS ADDTO(STC) ACID(opca region acid) PROCNAME(OPCA)
  ```

- Define a resource class for OPC/A in the RDT as follows:

  ```
  TSS ADDTO(RDT) RESCLASS(OPCCLASS) RESCODE(nn) ATTR(LONG)
  or
  TSS ADDTO(RDT) RESCLASS(IBMOPC) RESCODE(nn) ATTR(LONG)
  TSS REPLACE(RDT) RESCLASS(OPCCLASS) ACLST(ALL,UPDATE,READ,NONE)
  or
  TSS REPLACE(RDT) RESCLASS(IBMOPC) ACLST(ALL,UPDATE,READ,NONE)
  ```

  Failure to add this RDT entry will result in one of the following errors:

  TSS9410E   Unknown classname(OPCCLASS) for SVC(SVCNAME)

  ```
  TSS9410E     Unknown classname(IBMOPC) for SVC(SVCNAME)
  ```

- There are calls against predefined resource classes SURROGAT, APPLICATION, and JESJOBS. The resource entity used in the APPLICATION resource class is the OPC/A subsystem name, expressed as follows:

  ```
  TSS PERMIT(ACID) APPLICATION(opca subsystem name)
  ```

Note: In Step 3, the class names are derived from OPC/ESA documentation. Either class will work exactly the same, regardless of the release of CA Top Secret. If you are currently using OPC/ESA with a classname of OPCCLASS, there is no reason to change to the new classname of IBMOPC.

Lotus Domino GO Webserver

The Lotus Domino GO Webserver is an IBM offering that allows the MVS mainframe to act as an Internet web server. This offering is installed and managed as an OpenEdition application. UNIX Systems Services is the MVS implementation of a Unix environment. An ISPF interface known as the ISHELL is provided with OpenEdition and it is used to browse and edit files within this environment. By default, the Lotus Domino GO Webserver materials are installed within the OMVS environment in a directory named /usr/lpp/internet.

As documented by IBM, installation of the Internet Connection Server involves several steps involving the (RACF) security administrator.

Installation

If UNIX Systems Server has been previously setup, some or all of this step might be already complete. An ACID must exist for the OMVS, INETD, and TCPIP started tasks. In addition, these ACIDs must be connected to at least one OMVS group ACID.

The examples given below reflect default procedure names, typical group names, and typical GID values.

- Overall, these commands simply ensure that a valid OMVS UID and GID exist for each of the started tasks that access OMVS:

```
TSS CREATE(OMVSGRP) TYPE(GROUP)
                    NAME('DEFAULT OMVS GROUP')
                    DEPARTMENT(anydept)

TSS ADDTO(OMVSGRP) GID(1)

TSS CREATE(TTY) TYPE(GROUP)
                NAME('REQ''D OMVS TTY GROUP')
                DEPARTMENT(anydept)

TSS ADDTO(TTY) GID(2)

TSS CREATE(OMVS) TYPE(USER)
                 NAME('OPENMVS STC ID')
                 DEPARTMENT(anydept)
                 FACILITY(STC,APPC)
                 PASSWORD(NOPW,0)

TSS ADDTO(OMVS) UID(0)
                GROUP(OMVSGRP,TTY)
                DFLTGRP(OMVSGRP)

TSS CREATE(INETD) TYPE(USER)
                  NAME('OMVS INETD STC')
                  DEPARTMENT(anydept)
                  FACILITY(STC)
                  PASSWORD(NOPW,0)

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

TSS CREATE(TCPIP) TYPE(USER)
                  NAME('TCP/IP STC ID')
                  DEPARTMENT(anydept)
                  FACILITY(STC)
                  PASSWORD(NOPW,0)
TSS ADDTO(TCPIP) UID(0)
                 GROUP(OMVSGRP)
                 DFLTGRP(OMVSGRP)

TSS MODI OMVSTABS
TSS ADDTO(STC) PROCNAME(OMVS)
               ACID(OMVS)

TSS ADDTO(STC) PROCNAME(INETD)
               ACID(INETD)

TSS ADDTO(STC) PROCNAME(TCPIP)
               ACID(TCPIP)
```

- A TSS FACILITY should be created for the web server. Once created, this facility can be added to each user ACID that is allowed to logon to the web server. Tailor the command, and then add it to the existing CA Top Secret startup control options shown below.

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

- The Lotus Domino GO Webserver requires an ACID for the web server started task and for a web administrator. Both of these ACIDS must be connected to an OMVS Group ID for the web server. The commands shown below accomplish this using values following the IBM defaults.

  Note: The web server started task, whose procedure name is IMWEBSRV, is also referred to by IBM as the web server daemon.

  Changing the ID of the web administrator is also recommended; however, this change must be coordinated with updates to the web server configuration file.

  ```
  TSS CREATE(IMWEB) TYPE(GROUP)
                    NAME('WEBSERVER GROUP')
                    DEPARTMENT(anydept)

  TSS ADDTO(IMWEB) GID(205)

  TSS CREATE(WEBADM) TYPE(USER)
                     NAME('WEB ADMINISTRATOR')
                     DEPARTMENT(anydept)
                     FACILITY(IMWEB)
                     PASSWORD(password)

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

  TSS CREATE(WEBSRV) TYPE(USER)
                     NAME('WEBSERVER DAEMON/STC')
                     DEPARTMENT(anydept)
                     FACILITY(STC,IMWEB)
                     PASSWORD(NOPW,0)
  ```

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

- Three other user ACIDs, each having their own connected group, are required unless "surrogate user" support is disabled. This feature permits users to access the web server without requiring a signon. CA recommends that this feature be disabled for security reasons. To disable this feature, change the "Userid" option to "%%CLIENT%%" within the web server configuration file (see IBM documentation). If it is not disabled, the following commands will create ACIDs and groups for surrogate support following IBM examples:

```
TSS CREATE(EXTERNAL) TYPE(GROUP)
                     NAME('WEB GROUP')
                     DEPARTMENT(anydept)
TSS ADDTO(EXTERNAL) GID(999)
TSS CREATE(EMPLOYEE) TYPE(GROUP)
                     NAME('WEB GROUP')
                     DEPARTMENT(anydept)
TSS ADDTO(EMPLOYEE) GID(500)
TSS CREATE(SPECIAL) TYPE(GROUP)
                    NAME('WEB GROUP')
                    DEPARTMENT(anydept)
TSS ADDTO(SPECIAL) GID(255)
TSS CREATE(PUBLIC) TYPE(USER)
                   NAME('WEB SURROGATE ID')
                   DEPARTMENT(anydept)
                   FACILITY(IMWEB)
                   PASSWORD(NOPW,0)
TSS ADDTO(PUBLIC) UID(998)
                  GROUP(EXTERNAL)
                  DFLTGRP(EXTERNAL)
                  HOME(/) OMVSPGM(/bin/sh)
TSS CREATE(INTERNAL) TYPE(USER)
                     NAME('WEB SURROGATE ID')
                     DEPARTMENT(anydept)
                     FACILITY(IMWEB)
                     PASSWORD(NOPW,0)
TSS ADDTO(INTERNAL) UID(537)
                    GROUP(EMPLOYEE)
                    DFLTGRP(EMPLOYEE)
                    HOME(/)
                    OMVSPGM(/bin/sh)
```

```
TSS CREATE(PRIVATE) TYPE(USER)
                    NAME('WEB SURROGATE ID')
                    DEPARTMENT(anydept)
                    FACILITY(IMWEB)
                    PASSWORD(NOPW,0)
TSS ADDTO(PRIVATE) UID(416)
                   GROUP(SPECIAL)
                   DFLTGRP(SPECIAL)
                   HOME(/)
                   OMVSPGM(/bin/sh)
TSS MODI OMVSTABS
```

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

  ```
  TSS ADDTO(anydept) IBMFAC(BPX.)
  TSS PERMIT(WEBSRV) IBMFAC(BPX.DAEMON)
                     ACCESS(READ)
  TSS PERMIT(WEBSRV) IBMFAC(BPX.SERVER)
                     ACCESS(UPDATE)
  TSS ADDTO(anydept) SURROGAT(BPX.)
  TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.WEBADM)
                     ACCESS(READ)
  TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.PUBLIC)
                     ACCESS(READ)
  TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.PRIVATE)
                     ACCESS(READ)
  TSS PERMIT(WEBSRV) SURROGAT(BPX.SRV.INTERNAL)
                     ACCESS(READ)
  ```

- IBM documentation includes one installation step where RACF program control is discussed. In the first part of this step, all users are given READ access to three load libraries related to the web server. This part is easily translated as follows:

  ```
  TSS ADDTO(anydept) DSNAME(CEE.)
  TSS ADDTO(anydept) DSNAME(IMW.)
  TSS ADDTO(anydept) DSNAME(SYS1.)
  TSS PERMIT(ALL) DSNAME(CEE.V1R5M0.SCEERUN)
                  ACCESS(READ)
  TSS PERMIT(ALL) DSNAME(IMW.V1R1M0.IMWMOD1)
                  ACCESS(READ)
  TSS PERMIT(ALL) DSNAME(SYS1.LINKLIB)
                  ACCESS(READ)
  ```

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

(To RACF, these commands mark all programs in these libraries as NOPADCHK. To RACF this means that any program-restricted data set access should not have to list any of the programs from these libraries. In other words, this marks all programs from these libraries as being trusted, and therefore exempt, from any program accessed data set/PADS checks. These commands are not applicable to CA Top Secret.)

- A batch jobstream which contains all of the above commands is available from CA to replace the IBM supplied, RACF-based, install jobstreams named IMVJSEC, IMVJSECA, IMVRACLS, and IMVJPASS. The job can be found as member IMVSECUR within CA Top Secret CAI.SAMPJCL (file 9) in genlevel 9609 and above.
- Install steps which discuss permission bits and the "sticky bit" are related to OMVS file security itself and are unrelated to RACF. Therefore, such steps should be followed as described.

TONE

CA Top Secret comes with TONE, a product of Tone Software Corp., already defined as a default facility in the Facilities Matrix. This means that the security attributes that control CA Top Secret processing for TONE are predefined. These attributes, listed next, are actually suboptions of the FACILITY control option. A description of each of these suboptions is provided in the *Control Options Guide*.

Note: TONE is defined to CA Top Secret, through the Facilities Matrix, as a multi-user address space.

```
INITPGM=TON      id=T  TYPE=13
ATTRIBUTES=ACTIVE,SHRPRF,ASUBM,ABEND,MULTIUSER,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
```

Attributes are explained in the *Control Options Guide*.

If the default attributes are not applicable to your site, they can be changed in the Parameter File using the FACILITY control option.

Note: When TONE is active, it is supported as if it were the TSO subsystem.

VAM/SPF

CA Top Secret comes with VAM/SPF, a product of Boole & Babbage, Inc., already defined as a default facility in the Facilities Matrix. This means that the security attributes that control CA Top Secret processing for VAM/SPF are predefined. These attributes, listed next, are actually suboptions of the FACILITY control option. A description of each of these suboptions is provided in the *Control Options Guide*.

Note: VAM/SPF is defined to CA Top Secret through the Facilities Matrix with the name of VAMSPF, as a multi-user address space.

```
FACILITY DISPLAY FOR VAMSPF
INITPGM=VAM          id=V      TYPE=009
ATTRIBUTES=ACTIVE,SHRPRF,NOASUMB,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),INSTDATA,RNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,TSOC,LCFCMD
ATTRIBUTES=MSGLC,NOTRACE,NOEODINIT,IJU,NODORMPW,NONPWR
MODE=WARN DOWN=GLOBAL LOGGING=ACCESS,INIT,SMF,MSG,SEC9
UIDACID=8 LOCKTIME=000 DEFACID=*NONE*  KEY=8
MAXUSER=03000  PRFT=003
```

If the default attributes are not applicable to your site, they can be changed in the Parameter File using the FACILITY control option.

Appendix A: Converting SMU Control Statements to CA Top Secret

This section contains the following topics:

Overview

IMS internal security is defined by the Security Maintenance Utility (SMU). In IMS r10.1, SMU is not supported.

Prior to IMS r9.1, IMS provided functionality to replace many of the features of SMU. In IMS r9.1, IBM provided additional functionality to replace the remaining features of SMU with IMS internal mechanisms and external security calls. While running IMS 9.1, IMS installations need to migrate from any remaining SMU based definitions to these new facilities.

Important! You must be running IMS r9.1 to perform the conversion process.

Type 1 AOI Command Security Conversion

A type 1 AOI command is an IMS command issued by a transaction program using the IMS CMD communications call.

SMU provides security for type 1 AOI commands by controlling the commands issued by specific IMS transactions. SMU has two ways to define this security:

- A)(TCOMMAND control statement is defined for a command followed by CTRANS data statements for each transaction that can issue the command.
- A)(CTRANS control statement is defined for a transaction followed by TCOMMAND data statements for each command issued by the transaction.

In both cases the statements establish a relationship between IMS commands and the transactions that issue them.

The security provided by the CA Top Secret IMS security interface for a type 1 AOI command is determined by the AOI= parameter in the IMS system definition TRANSACT macro for the transaction issuing the command. Possible values are:

NO

The transaction is not allowed to issue any AOI type 1 commands. This restriction is enforced by IMS. No calls are made to CA Top Secret.

YES

Validations are performed for AOI type 1 commands using the ACID of the user who entered the transaction issuing the command. Checking user access to the command is the most granular security option. An individual user or a set of users is given access to a command. The users can be given access to one command entered through a transaction and denied access to another command entered through the same transaction. This alternative allows for individual accountability in the security process; when an access is denied, the violation is for the individual user.

The migration from transaction based to user based security is not straightforward. If a user has access to a transaction that issues type 1 AOI commands they must also be given access to each command the transaction issues.

This alternative uses the TSSISMU3 conversion program to migrate from SMU AOI security to CA Top Secret.

TRAN

CA Top Secret performs security validations for AOI type 1 commands using an ACID defined for the transaction issuing the command. This does not provide individual accountability. If access is denied the violation is for the transaction ACID rather than the user executing the transaction.

Select this option only if CA Top Secret is used for transaction security. CA Top Secret controls user access to the transaction using IMS transaction security and controls what commands the transaction can issue using AOI command security. This alternative is a direct parallel to the SMU security process and migration is straightforward.

This alternative uses the TSSISMU1 conversion program to migrate from SMU AOI security to CA Top Secret.

CMD

CA Top Secret performs security validations for the transaction issuing the AOI type 1 command using an ACID of the command. This alternative does not provide individual accountability. If access is denied the violation is for the ACID for the command.

Select this option only if CA Top Secret is used for transaction security. CA Top Secret controls user access to the transaction using IMS transaction security based on the user and what commands the transaction can issue using transaction security based on the command. This alternative is an inverse of the SMU security process.

No conversion program is provided for this alternative.

Note: Mixed AOI security implementation is possible in a single IMS region. Specify AOI=YES is for some transactions and AOI=TRAN for others.

The TSSISMU1 Utility

This TSSISMU1 utility is a direct conversion of the SMU security process.

The TSSISMU1 utility processes the SMU statements. For each transaction in the SMU AOI security statements, the utility generates a TSS CREATE command to create an ACID for the transaction. For each command/transaction combination in the SMU statements, the utility generates a TSS PERMIT statement to permit the transaction ACID access to the command. If the command specified in a TCOMMAND statement is * TSS PERMIT statements are generated for each IMS command eligible for AOI.

AOI=TRAN must be specified in the IMS system definition (IMS stage 1) in the TRANSACT macro for any transaction that issues AOI type 1 commands. The utility optionally issues a report listing the transactions that require AOI=TRAN in the IMS system definition.

The utility can also optionally take as input a data set with the IMS system definition (IMS stage 1) statements containing the transaction definitions, and generate an updated set of statements that includes the AOI=TRAN parameter for the transactions that require it. The AOI=TRAN parameter is unconditionally added to the TRANSACT macro definition for any transactions that issue AOI type 1 commands, even if the TRANSACT macro already contains an AOI parameter specification. If two AOI parameters are specified on a TRANSACT macro, an error is generated during the IMS stage 1 process, allowing you to determine the correct specification.

Examples: TSSISMU1 Conversion

In this example the SMU input is one of the following:

```
)( TCOMMAND DISPLAY
CTRANS TRANAOI
```

```
)( CTRANS TRANAOI
TCOMMAND DISPLAY
```

The utility generates:

```
TSS CREATE(TRANAOI) TYPE(USER) NAME('AOI transaction')
                    PASSWORD(TRANAOI)
                    SOURCE(TRANAOI)
                    NOSUSPEND
                    DEPT(department)
                    FAC(imsfac)
TSS PERMIT(TRANAOI) CIMS(DIS)
                    FAC(imsfac)
```

In this example the SMU input is one of the following:

```
)( TCOMMAND *
CTRANS TRANAOI
```

```
)( CTRANS TRANAOI
 TCOMMAND *
```

The utility generates PERMIT statements for every IMS command eligible for AOI:

```
TSS CREATE(TRANAOI) TYPE(USER)
                    NAME('AOI transaction')
                    PASSWORD(TRANAOI)
                    SOURCE(TRANAOI)
                    NOSUSPEND
                    DEPT(department)
                    FAC(imsfac)
TSS PERMIT(TRANAOI) CIMS(ACT)
                    FAC(imsfac)
TSS PERMIT(TRANAOI) CIMS(ALL)
                    FAC(imsfac)
TSS PERMIT(TRANAOI) CIMS(ASS)
                    FAC(imsfac)
```

TSSISMU1 JCL Sample

The following sample JCL executes the TSSISMU1 conversion utility:

```
//    JOB …
//TSSISMU1     EXEC PGM=TSSISMU1,PARM='department,imsfac'
//SMU          DD   DSN=smu.input,DISP=SHR
//TSSCMDS      DD   DSN=tsscmds.output,DISP=SHR
//REPORT       DD   SYSOUT=*
//STG1IN       DD   DSN=ims.stage1.input,DISP=SHR
//STG1OUT      DD   DSN=updated.ims.stage1,DISP=SHR
//
```

Department

Specifies the input parameter used in the TSS CREATE commands to create ACIDs for the transaction names. The transaction ACIDs are defined in the specified department.

Imsfac

Specifies the input parameter for the facility name of the IMS control region whose SMU input is converted. This value is used in the TSS CREATE commands for the transaction ACIDs to give the ACIDs access to the IMS region and in the TSS PERMIT commands to give the transaction access to the command in the IMS region.

SMU

This DD statement specifies the input data set containing the SMU statements for the IMS region being processed.

This must be a sequential data set with a record length of 80.

TSSCMDS

This DD statement specifies the output data set created by the TSSISMU1 utility containing the TSS statements generated by the conversion utility.

This must be a sequential data set with a record length of 80.

REPORT

(Optional) This DD statement specifies the output data set created by the TSSISMU1 utility containing a report listing the transactions in the IMS system definition (IMS stage 1) that require the AOI=TRAN parameter.

If the DD statement specifies a data set rather than SYSOUT, the data set must be sequential with a record length of 80.

If not specified the report is not generated.

STG1IN

(Optional) This DD statement specifies the input data set containing the IMS system definition (IMS stage 1) statements that include the TRANSACT macro statements for all transactions.

The data set must be sequential with a record length of 80.

If not specified the IMS system definition update processing is not performed. If this DD statement is specified, the STG1OUT DD statement is required.

The utility does not expand COPY statements or require the entire IMS system definition. The STG1IN input data set must contain the IMS system definition TRANSACT macro statements for all transactions.

STG1OUT

This DD statement specifies the output data set created by the TSSISMU1 utility, containing the updated IMS system definition (IMS stage 1) statements. The AOI=TRAN parameter is added to the TRANSACT macro definition for any transactions that issue AOI type 1 commands. The data set must be a sequential data set, with a record length of 80.

This DD statement is optional. It is only required if the STG1IN DD statement is specified.

Notes:

- The STG1OUT DD statement should not point to the same data set as the STG1IN DD statement.
- The AOI=TRAN parameter is unconditionally added to the TRANSACT macro definition for any transactions that issue AOI type 1 commands, even if the TRANSACT macro already contains an AOI parameter specification. If two AOI parameters are specified on a TRANSACT macro, an error is generated during the IMS stage 1 process allowing you to determine the correct specification.

How TSSISMU1 Processing Works

The TSSISMU1 conversion utility:

- Parses the department and facility values from the PARM= execution parameter in the TSSISMU1 job stream
- Processes the SMU information from the data set specified in the SMU DD statement in the execution JCL.
- Identifies the)(TCOMMAND control statements and extracts the command name from the control statement

  If the command name in the TCOMMAND statement is longer than three characters, the utility replaces it with the first three characters of the command verb.

  - Processes the CTRANS data statements that follow the)(TCOMMAND control statement.
  - (If this is the first time the transaction has been processed.) Generates a TSS CREATE command to create an ACID for the transaction for each transaction.

    The department for the ACID is the department specified in the PARM= parameter in the utility JCL, and the facility for the ACID is the facility in the PARM= parameter.
  - Generates a TSS PERMIT statement to permit access by the transaction ACID to the command.

    If the command specified in the TCOMMAND statement is * TSS PERMIT statements are generated for each IMS command eligible for AOI.
- Identifies the)(CTRANS control statements and extracts the transaction name from the control statement.
  - (If this is the first time the transaction is processed.) Generates a TSS CREATE command to create an ACID for the transaction.

    The department for the ACID is the department specified in the PARM= parameter in the utility JCL, and the facility for the ACID is the facility in the PARM= parameter.
  - Processes the TCOMMAND data statements that follow the)(CTRANS control statement. For each TCOMMAND data statement, the utility extracts the command name.

    If the command name in the TCOMMAND statement is longer than three characters, the utility replaces it with the first three characters of the command verb.
  - Generates the TSS PERMIT statement to permit access by the transaction ACID to the command.

    If the command specified in a TCOMMAND statement is * TSS PERMIT statements are generated for each IMS command eligible for AOI.

- (If the REPORT DD statement is specified in the utility JCL.) Generates a report listing all transactions identified in the SMU conversion process as issuing type 1 AOI commands.

  These transactions require the AOI=TRAN parameter in the TRANSACT macro transaction definitions in the IMS system definition (IMS stage 1) statements.

  Use this report to ensure that the appropriate transaction definitions are updated.

- (If the STG1IN and STG1OUT DD statements are specified in the utility JCL.)
  - Copies the IMS system definition statements from the STG1IN data set to the STG1OUT data set.
  - Identifies the TRANSACT macros for a transaction identified in the SMU conversion process as issuing AOI type 1 commands and adds an AOI=TRAN parameter to the end of the TRANSACT macro definition.

TSS Command Execution

Review the TSS commands generated by the TSSISMU1 utility for accuracy before you execute them.

This sample JCL executes the TSS statements in a batch TSO job:

```
//    JOB …
//IKJEFT01    EXEC  PGM=IKJEFT01,REGION=0M
//SYSTSPRT    DD    SYSOUT=*
//SYSPRINT    DD    SYSOUT=*
//SYSTSIN     DD    DSN=tsscmds.input,DISP=SHR
//
```

To execute the TSS statements the user must have an ACID with sufficient authority to create the transaction ACIDs and perform the command PERMITs.

Transaction Based CA Top Secret for AOI Commands Implementation

To implement AOI=TRAN CA Top Secret for AOI commands:

- Determine the department the transaction ACIDs is created in.
- Determine the facility of the IMS region whose SMU security is being converted.
- Run the TSSISMU1 conversion utility program using the SMU statements for the IMS region.
- Review the TSS command statements created by the TSSISMU1 utility for accuracy.
- Execute the TSS commands to update CA Top Secret.
- Change the IMS system definition (IMS stage 1) for the IMS region to include the AOI=TRAN parameter in the TRANSACT macro for all transactions that issue AOI type 1 commands. Use the report generated by the TSSISMU1 utility to manually change the system definition. Use the utility stage 1 update process to automatically update the system definition.
- Run the IMS system definition.
- Change the IMS execution parameter to AOI1=R.
- Restart IMS.

The TSSISMU3 Utility

Use the TSSISMU3 utility to convert the transaction-based SMU security to a user-based security policy for the commands.

The TSSISMU3 utility processes the SMU statements. For each transaction in the SMU statements that can issue AOI type 1 commands, the utility determines who has access to the transaction. For each command the transaction can issue, the utility generates PERMIT statements to permit access to the command for every user that has access to the transaction. If the command specified in a TCOMMAND statement is * TSS PERMIT statements are generated for each IMS command eligible for AOI.

The TSS PERMIT statements generated by this utility require that AOI=YES be specified in the IMS system definition (IMS stage 1) in the TRANSACT macro for any transaction that issues AOI type 1 commands.

The utility optionally issues a report listing the transactions require AOI=YES in the IMS system definition (IMS stage 1).

The utility can optionally take as input a data set with the IMS system definition (IMS stage 1) statements containing the transaction definitions, and generate an updated set of statements that includes the AOI=YES parameter for the transactions that require it. The AOI=YES parameter is unconditionally added to the TRANSACT macro definition for any transactions that issue AOI type 1 commands, even if the TRANSACT macro already contains an AOI parameter specification. If two AOI parameters are specified on a TRANSACT macro, an error is generated during the IMS stage 1 process, allowing you to determine the correct specification.

Note: The utility processes a TSSCFILE data set containing the IMS transaction security policy defined in the OTRAN and TIMS resource classes, and uses it to create the security policy for the corresponding type 1 AOI commands. The utility does not support transaction security policy in the LCF resource class. If transaction security policy is defined in the LCF resource class, this utility cannot be used as a migration aid in the SMU conversion.

Examples: TSSISMU3 conversion

In this example, the SMU input is one of the following:

```
)( TCOMMAND DISPLAY
 CTRANS TRANAOI
```

```
)( CTRANS TRANAOI
 TCOMMAND DISPLAY
```

The users USER01 and USER02 have access to the TRANAOI transaction. The utility generates:

```
TSS PERMIT(USER01) CIMS(DIS)
                   FAC(imsfac)
TSS PERMIT(USER02) CIMS(DIS)
                   FAC(imsfac)
```

In this example, the SMU input is one of:

```
)( TCOMMAND *
 CTRANS TRANAOI

)( CTRANS TRANAOI
 TCOMMAND *
```

The users USER01 and USER02 have access to the TRANAOI transaction. The utility generates PERMIT statements for each user for every IMS command eligible for AOI:

```
TSS PERMIT(USER01) CIMS(ACT) FAC(imsfac)
TSS PERMIT(USER01) CIMS(ALL) FAC(imsfac)
TSS PERMIT(USER01) CIMS(ASS) FAC(imsfac)
. . .
TSS PERMIT(USER02) CIMS(ACT) FAC(imsfac)
TSS PERMIT(USER02) CIMS(ALL) FAC(imsfac)
TSS PERMIT(USER02) CIMS(ASS) FAC(imsfac)
. . .
```

TSSISMU3 Authorization Requirements

There are two steps to the TSSISMU3 utility process.

- Execute the TSSCFILE utility to obtain the current security policy for IMS transactions. This step should be executed by the MSCA to ensure that all policy information is obtained.
- Execute the TSSISMU3 utility. There are no authorization requirements for the TSSISMU3 utility.

TSSCFILE Sample JCL

The following sample JCL executes the TSSCFILE utility:

```
//    JOB …
//TSSCFILE     EXEC PGM=TSSCFILE
//OUT          DD   DSN=tsscfile.output,
//             DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,
//             VOL=SER=vvvvvv,
//             SPACE=(CYL,(10,10),RLSE),
//             DCB=(RECFM=FB,LRECL=300)
//PRINT        DD   SYSOUT=*
//IN           DD   *
TSS WHOHAS OTRAN(*) DATA(MASK)
TSS WHOHAS TIMS(*) DATA(MASK)
//
```

OUT

This DD statement specifies the output data set created by the TSSCFILE utility containing the output of the TSS WHOHAS commands. This data set is used as input to the TSSISMU3 utility.

This must be a sequential data set with a record length of 300.

PRINT

This DD statement specifies the utility report data set created by the TSSCFILE utility.

IN

This DD statement specifies the data set containing the TSS WHOHAS commands. These commands are required and must be entered exactly as shown.

TSSISMU3 Sample JCL

This example JCL executes the TSSISMU3 conversion utility:

```
//    JOB …
//TSSISMU3   EXEC PGM=TSSISMU3,PARM=imsfac
//TSSCFILE   DD  DSN=tsscfile.input,DISP=SHR
//SMU        DD  DSN=smu.input,DISP=SHR
//TSSCMDS    DD  DSN=tsscmds.output,DISP=SHR
//REPORT     DD  SYSOUT=*
//STG1IN     DD  DSN=ims.stage1.input,DISP=SHR
//STG1OUT    DD  DSN=updated.ims.stage1,DISP=SHR
//
```

Imsfac

Specifies the facility name of the IMS control region whose SMU input is being converted. This value is used in the TSS PERMIT statements generated to give access to the command in the IMS region.

TSSCFILE

This DD statement specifies the input data set containing the TSSCFILE input for transaction access.

This must be a sequential data set with a record length of 300.

SMU

This DD statement specifies the input data set containing the SMU statements for the IMS region being processed.

This must be a sequential data set with a record length of 80.

TSSCMDS

This DD statement specifies the output data set created by the TSSISMU3 utility containing the TSS statements generated by the conversion utility.

This must be a sequential data set with a record length of 80.

REPORT

(Optional) This DD statement specifies the output data set created by the TSSISMU3 utility containing a report listing the transactions in the IMS system definition (IMS stage 1) that require the AOI=YES parameter.

If the DD statement specifies a data set rather than SYSOUT, the data set must be sequential with a record length of 80.

If the DD statement is not specified the report is not generated.

STG1IN

(Optional) This DD statement specifies the input data set containing the IMS system definition (IMS stage 1) statements that include the TRANSACT macro statements for all transactions.

This must be a sequential data set with a record length of 80.

If the DD statement is not specified, the IMS system definition update processing is not performed. If this DD statement is specified, the STG1OUT DD statement is required.

Note: The utility does not expand COPY statements or require the entire IMS system definition. The STG1IN input data set must contain the IMS system definition TRANSACT macro statements for all transactions.

STG1OUT

(Optional) This DD statement specifies the output data set created by the TSSISMU3 utility containing the updated IMS system definition (IMS stage 1) statements. The AOI=YES parameter is added to the TRANSACT macro definition for any transactions that issue AOI type 1 commands.

This must be a sequential data set with a record length of 80.

This DD statement is only required if the STG1IN DD statement is specified.

Notes:

- The STG1OUT DD statement should not point to the same data set as the STG1IN DD statement.
- The AOI=YES parameter is unconditionally added to the TRANSACT macro definition for any transactions that issue AOI type 1 commands, even if the TRANSACT macro already contains an AOI parameter specification. If two AOI parameters are specified on a TRANSACT macro, an error is generated during the IMS stage 1 process, allowing you to determine the correct specification.

How TSSISMU3 Processing Works

The TSSISMU3 conversion utility:

- Parses the facility value from the PARM= execution parameter in the TSSISMU3 job stream
- Processes the information from the TSSCFILE input data set containing the transaction access information.
- Uses the TSSCFILE information to create an in-storage table of all transactions and who has access to them.
- Processes the SMU information from the data set specified in the SMU DD statement in the execution JCL.
- Identifies)(TCOMMAND control statements and extracts the command name from the control statement .

  If the command name in the TCOMMAND statement is longer than three characters, the utility replaces it with the first three characters of the command verb.

  - Processes the CTRANS data statements that follow the)(TCOMMAND control statement.
  - Searches the in-storage TSSCFILE table to determine who has access to the transaction for each transaction identified in a CTRANS data statement.
  - Generates a PERMIT statement giving the ACID access to the command from the facility for the IMS region for each ACID with access to the transaction.

    If the command specified in the TCOMMAND statement is * TSS PERMIT statements are generated for each IMS command eligible for AOI.
- Identifies)(CTRANS control statements and extracts the transaction name from the control statement.
- Processes the TCOMMAND data statements that follow the)(CTRANS control statement.
- Extracts the command name for each TCOMMAND data statement.

  If the command name in the TCOMMAND statement is longer than three characters, the utility replaces it with the first three characters of the command verb.
- Searches the in-storage TSSCFILE table to determine who has access to the transaction.
- Generates a PERMIT statement giving the ACID access to the command from the facility for the IMS region for each ACID with access to the transaction.
- (If the command specified in a TCOMMAND statement is *) generates TSS PERMIT statements for each IMS command eligible for AOI.

- If the REPORT DD statement is specified in the utility JCL, generates a report listing all transactions identified in the SMU conversion process as issuing type 1 AOI commands.

  These transactions require the AOI=YES parameter in the TRANSACT macro transaction definitions in the IMS system definition (IMS stage 1) statements.

  Use this report to ensure that the appropriate transaction definitions are updated.

- Provide an updated IMS system definition if the STG1IN and STG1OUT DD statements are specified in the utility JCL.

  The utility copies the IMS system definition statements from the STG1IN data set to the STG1OUT data set. When it identifies a TRANSACT macro for a transaction identified in the SMU conversion process as issuing AOI type 1 commands, it adds an AOI=YES parameter to the end of the TRANSACT macro definition.

TSS Command Execution

Review the TSS commands generated by the TSSISMU3 utility for accuracy before you execute them.

This sample JCL executes the TSS statements in a batch TSO job.

```
//    JOB …
//IKJEFT01   EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT   DD  SYSOUT=*
//SYSPRINT   DD  SYSOUT=*
//SYSTSIN    DD  DSN=tsscmds.input,DISP=SHR
//
```

To execute the TSS statements, the user must have an ACID with sufficient authority to perform the command PERMITs.

User Based CA Top Secret for AOI Commands Implementation

To implement AOI=YES CA Top Secret for AOI commands:

- Determine the facility of the IMS region whose SMU security is being converted.
- Run the TSSCFILE utility to obtain the current security policy for IMS transactions.
- Run the TSSISMU3 conversion utility program using the TSSCFILE input and the SMU statements for the IMS region.
- Review the TSS command statements created by the TSSISMU3 utility for accuracy.
- Execute the TSS commands to update CA Top Secret.
- Change the IMS system definition (IMS stage 1) for the IMS region to include the AOI=YES parameter in the TRANSACT macro for all transactions that issue AOI type 1 commands. The report generated by the TSSISMU3 utility is used to manually change the system definition, or the utility stage 1 update process is used to automatically update the system definition.
- Run the IMS system definition.
- Change the IMS execution parameter to AOI1=R.
- Restart IMS

AGN Security to RAS Conversion

An application group name (AGN) identifies a group of transactions, terminals, and PSBs that an IMS dependent region, such as a BMP, is permitted to access. AGNs are defined for an IMS system using the Security Maintenance Utility (SMU).

This example shows how an AGN is defined in the SMU input:

```
)(  AGN      name
    AGLTERM  terminal name
    AGPSB    PSB name
    AGTRAN   transaction
```

name

Specifies the name selected for the AGN. This statement identifies the terminals, PSBs, and transactions that follow it as a group that restricts access by dependent regions.

Each AGN name must be unique.

Range: 8 characters or less

terminal name

Specifies the logical terminal (LTERM) name for a terminal included in the AGN.

PSB name

Specifies the name of a PSB included in the AGN.

transaction

Specifies the name of a transaction included in the AGN.

An IMS dependent region requests the use of an AGN by specifying the AGN name in the startup JCL for the dependent region. IMS performs a security validation to verify that the dependent region is permitted to use the AGN. The IMS control region then limits the dependent region's use of IMS resources to those defined in the AGN that was requested.

Resource Access Security (RAS)

IMS r9.1 introduced RAS as a replacement for AGNs and AGN security. In IMS r10.1, AGNs and AGN security is removed.

With RAS, IMS performs a direct security validation of the terminals, PSBs, and transactions that a dependent region can access. For example, before a transaction is scheduled to execute in a message processing regions, IMS performs a security validation to see if the logonid of the message processing region is allowed to execute the transaction. If not, the transaction is not scheduled in that message processing region.

AGN Security to RAS with CA Top Secret Conversion

Before converting AGN definitions to RAS security, evaluate whether you need this level of security. Many IMS installations implemented AGN security to secure the use of PSBs in IMS BMP regions. Because of its design, when AGN security is implemented, it must be implemented for all IMS dependent regions. This lead to the definition of AGNs for message process regions that look like this:

```
)(   AGN      MSG1
     AGLTERM  ALL
     AGPSB    ALL
     AGTRAN   ALL
```

These AGNs are designed to prevent AGN security, implemented for PSB security in BMPs, from interfering in the processing of message processing regions.

If this matches your implementation evaluate whether existing PSB security in the CA Top Secret IMS interface is sufficient for your security requirements and if AGN security can be eliminated rather than converted.

The TSSISMU2 utility is used to migrate AGN security to RAS.

The TSSISMU2 Utility

Use the TSSISMU2 utility to simplify the conversion from AGN security to RAS. The utility:

- Processes the SMU control statements for the IMS region being converted.
- Determines what ACIDs have access to the AGN for each AGN defined in the SMU input.
- Generates an ADDTO statement to give ownership of the resource for each element assigned to the AGN in the SMU input, if the resource is not currently owned in CA Top Secret.
- Generates PERMIT statements to give access to the element to the same set of users that have access to the AGN. This allows the same access to the AGN element that was held on the AGN.

If the element for the AGN is ALL, the utility generates an ADDTO statement giving ownership of the *ALL* resource to the MSCA, and generates a PERMIT statement with a resource entity of *ALL*.

For information on the use of the *ALL* value in a PERMIT statement , see the *Command Functions Guide*.

Examples: TSSISMU2 conversion

In this example, the TSSISMU2 utility is executed with PARM='MASTER,IMSDEPT,IMSPROD', and the SMU input is:

```
)( AGN BMP1
  AGPSB PGMBMP1
```

The users USER01 and USER02 have access to the BMP1 AGN. The utility generates:

```
TSS ADDTO(IMSDEPT) IIMS(PGMBMP)
TSS PERMIT(USER01) IIMS(PGMBMP)
                   FAC(IMSPROD)
TSS PERMIT(USER02) IIMS(PGMBMP)
                   FAC(IMSPROD)
```

In this example, the TSSISMU2 utility is executed with PARM='MASTER,IMSDEPT,IMSPROD', and the SMU input is:

```
)( AGN MSG1
  AGPSB ALL
```

The users USER01 and USER02, have access to the MSG1 AGN. The utility generates:

```
TSS ADDTO(MASTER) IIMS(*ALL*)
TSS PERMIT(USER01) IIMS(*ALL*)
                   FAC(IMSPROD)
TSS PERMIT(USER02) IIMS(*ALL*)
                   FAC(IMSPROD)
```

Authorization requirements for the TSSISMU2 utility

There are two steps to the TSSISMU2 utility process.

- Execute the TSSCFILE utility to obtain the current resource ownership information for IMS PSBs, transactions, and LTERMs and the current security policy for AGNs. This step should be executed by the MSCA to ensure that all security information is obtained.
- Executes the TSSISMU2 utility. There are no authorization requirements for the TSISMU2 utility.

TSSCFILE Sample JCL

This sample JCL executes the TSSCFILE utility:

```
//    JOB …
//TSSCFILE      EXEC PGM=TSSCFILE
//OUT           DD   DSN=tsscfile.output,
//              DISP=(NEW,CATLG,DELETE),
//              UNIT=SYSDA,
//              VOL=SER=vvvvvv,
//              SPACE=(CYL,(10,10),RLSE),
//              DCB=(RECFM=FB,LRECL=300)
//PRINT    DD    SYSOUT=*
//IN       DD    *
TSS WHOOWNS IIMS(*)
TSS WHOOWNS TIMS(*)
TSS WHOOWNS LIMS(*)
TSS WHOHAS APPL(*) DATA(MASK)
//
```

OUT

This DD statement specifies the output data set created by the TSSCFILE utility containing the output of the TSS WHOOWNS and WHOHAS commands. This data set is used as input to the TSSISMU3 utility.

This must be a sequential data set with a record length of 300.

PRINT

This DD statement specifies the utility report data set created by the TSSCFILE utility.

IN

This DD statement specifies the data set containing the TSS CFILE input commands. These commands are required and must be entered exactly as shown.

TSSISMU2 Sample JCL

This sample JCL executes the TSSISMU2 conversion utility:

```
//    JOB …
//TSSISMU2   EXEC PGM=TSSISMU2,PARM='msca,owner,imsfac'
//TSSCFILE   DD  DSN=tsscfile.input,DISP=SHR
//SMU        DD  DSN=smu.input,DISP=SHR
//TSSCMDS    DD  DSN=tsscmds.output,DISP=SHR
//
```

msca

Specifies the ACID of the CA Top Secret MSCA. When an AGN element of ALL is processed, this value is used in the ADDTO statement giving ownership of the *ALL* resource to the MSCA.

owner

Specifies the ACID used in the ADDTO statement giving ownership of any AGN element resources not already owned. If different owning ACIDs are desired for different AGN element resources, a placeholder literal such as ???????? can be specified. The TSSCMDS data set can then be edited, and the correct values for the owning ACIDs substituted before the commands are executed.

imsfac

Specifies the facility name of the IMS control region whose SMU input is being converted. This value is used in the TSS PERMIT statements generated to give access to the AGN elements in the IMS region.

TSSCFILE

This DD statement specifies the input data set containing the TSSCFILE input for AGN access and AGN element ownership.

This must be a sequential data set with a record length of 300.

SMU

This DD statement specifies the input data set containing the SMU statements for the IMS region being processed.

This must be a sequential data set, with a record length of 80.

TSSCMDS

This DD statement specifies the output data set created by the TSSISMU2 utility, containing the TSS statements generated by the conversion utility.

This must be a sequential data set with a record length of 80.

How TSSISMU2 Processing Works

The TSSISMU2 conversion utility:

- Parses the MCSA, owner, and facility values from the PARM= execution parameter in the TSSISMU2 job stream
- Processes the information from the TSSCFILE input data set to create storage tables of AGN access and AGN element resource ownership
- Processes the SMU information from the data set specified in the SMU DD statement in the execution JCL.
- Identifies AGN control statements and extracts the AGN name from the control statement
- Processes the AGPSB, AGTRAN, and AGLTERM data statements that follow the AGN control statement
  - Extracts the AGN element name from the data statement for each AGN element
  - Scans the resource ownership tables created in the TSSCFILE process to determine if the AGN element resource is owned.
  - Generates an ADDTO statement to give resource ownership to unowned resources

    If the AGN element is ALL, the ACID in the ADDTO statement is the value of the MCSA operand in the PARM= execution parameters. If the AGN element is a specific resource, the ACID in the ADDTO statement is the value of the owner operand in the PARM= execution parameters.
- Processes the AGN access table to generate the security policy for the AGN element.
- Generates a PERMIT statement giving the ACID access to the AGN element from the facility of the IMS region for each ACID that has access to the AGN.

  If the AGN element is ALL, the PERMIT statement is for the resource entity *ALL*, giving access to all resources

Executing the TSS Commands

Review the TSS commands generated by the TSSISMU2 utility for accuracy before you execute them.

This sample JCL executes the TSS statements in a batch TSO job:

```
//    JOB …
//IKJEFT01     EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT     DD  SYSOUT=*
//SYSPRINT     DD  SYSOUT=*
//SYSTSIN      DD  DSN=tsscmds.input,DISP=SHR
//
```

The user must have an ACID with sufficient authority to perform the command PERMITs.

Migrating AGN Security to RAS Security

To migrate from AGN security to RAS security:

- Determine whether you want to perform RAS security. If the existing CA Top Secret PSB security is sufficient for your security requirements, a complete AGN to RAS conversion may not be necessary.
- Decide if you want to assign resource ownership to a single ID, or use a substitution literal in the owner value of the PARM= execution parameters.
- Determine the facility of the IMS region whose SMU security is being converted.
- Run the TSSCFILE utility to obtain the security policy for access to AGNs, and the resource ownership information for the AGN element resource classes.
- Run the TSSISMU2 conversion utility program using the TSSCFILE input and the SMU statements for the IMS region being converted.
- Review the TSS command statements created by the TSSISMU2 utility for accuracy.
- Execute the TSS commands to update CA Top Secret.
- Change the IMS system definition (IMS stage 1) parameters to stop performing AGN security and start performing RAS security.
- Run the IMS system definition.
- Change the IMS execution parameter to ISIS=R.
- Restart IMS.

Terminal Based Security to CA Top Secret Conversion

SMU provides security for transactions and commands by identifying the terminals at which the transactions and commands are entered.

SMU has two ways to define this security:

- A)(TERMINAL control statement is defined for an IMS logical terminal (LTERM), followed by TRANSACT and COMMAND data statements for the transactions and commands that can entered at that terminal.
- A)(TRANSACT or)(COMMAND control statement is defined for a transaction or command, followed by TERMINAL data statements for each terminal that can enter the transaction or command.

In both cases the statements establish a relationship between IMS LTERMs and the IMS commands and transactions that can be entered at the terminals.

The CA Top Secret IMS interface provides security for IMS transactions and commands entered at a terminal.

To provide user-based security in CA Top Secret for an IMS transaction or command, a PERMIT is issued for the transaction or command permitting access to a specific user or to a profile. This is not a direct parallel to SMU terminal-based security. Because it is based on user identification and user-based security policy, it is the best practice implementation for IMS security, and should be used unless it is impractical or impossible.

For terminals that cannot or will not sign on, CA Top Secret provides a form of terminal based security for IMS transactions and commands called automatic terminal signon, or ATS. With ATS, an ACID is created for the terminal name. The ACID is used to permit access to IMS transactions and commands entered at the terminal. Because it is terminal based, this alternative does not provide user-level granularity in security policy. It should only be used if a terminal user cannot sign on, it is not important who the terminal user is, or if the physical security for the IMS terminal is sufficient to determine who the terminal user is.

SMU Terminal Based Security to CA Top Secret Conversion

If you have IMS regions that have SMU terminal security definitions, evaluate whether you need to convert the SMU security definitions to CA Top Secret, and whether the CA Top Secret should be user-based or terminal-based.

If you are converting the SMU terminal security for an IMS region that also uses the CA Top Secret IMS interface for command and transaction security, you already have security policy defined for your IMS transactions and commands. Examine the CA Top Secret policy to determine if the SMU terminal-based security definitions are redundant and can be eliminated.

If you are converting an IMS region that does not already use the CA Top Secret IMS security interface, or if CA Top Secret security for transactions or commands has been disabled in the IMS region, you probably do not have security policy defined for your IMS transactions and commands. The best practice implementation for CA Top Secret for IMS transactions and commands is a user-based security implementation. Determine who your IMS users are and what IMS transactions and commands they should have access to, and write your security policy accordingly. This is a manual process and the SMU terminal security definitions are usually of little help. If you decide on a terminal-based security implementation using automatic terminal signon, the SMU terminal security definitions provide the roadmap for the security policy.

Use the TSSISMU4 conversion program to migrate from SMU terminal security to CA Top Secret.

The TSSISMU4 Utility

Use the TSSISMU4 utility to provide a skeleton for the security policy in a conversion from SMU terminal security to CA Top Secret. This security policy can be for either a user-based or terminal-based security implementation.

The utility processes the SMU security definitions for the IMS region being converted.

If the conversion is to a user-based security policy:

- For each unique transaction or command in a SMU terminal security definition, the utility checks if the resource is currently owned in CA Top Secret.
- If the resource is not owned in CA Top Secret, the utility generates an ADDTO statement to give ownership of the resource.
- For each terminal/transaction or terminal/command combination in the SMU definitions, the utility generates a PERMIT statement for the IMS transaction or command, with a dummy value in the ACID initialized to the IMS logical terminal name.
- Determine who the users are who require access to the transaction or command, and change the ACID value in the PERMIT statement to reflect those users.
- If necessary, the PERMIT statement can be duplicated to permit access from different sets of users.

If the conversion is to a terminal-based security policy based on automatic terminal signon:

- For each unique terminal in the SMU terminal security definition, the utility generates a CREATE statement to create an ACID for the IMS logical terminal name.
- For each transaction or command in a SMU terminal security definition, the utility checks if the resource is currently owned in CA Top Secret.
- If the resource is not owned in CA Top Secret, the utility generates an ADDTO statement to give ownership of the resource.
- For each terminal/transaction or terminal/command combination in the SMU definitions, the utility generates a PERMIT statement to permit the LTERM ACID access to the IMS transaction or command.
- Determine what IMS physical terminal name corresponds to the LTERM name in the SMU definition. This physical terminal name is obtained from the IMS system definitions (IMS stage 1). Then change the LTERM value in the CREATE and PERMIT statements to the physical terminal name.

Examples: TSSISMU4 conversion

In this example, the TSSISMU4 utility is executed with PARM='ATS,OWNUSER,IMSPROD,IMSDEPT', and the SMU input is:

```
)( TERMINAL TERM1
  TRANSACT PAYINQ
  COMMAND DISPLAY
```

The utility generates the following PERMIT commands:

```
TSS CREATE(TERM1) TYPE(USER)
                  NAME('ATS terminal')
                  PASSWORD(TERM1)
                  SOURCE(TERM1)
                  NOSUSPEND
                  DEPT(IMSDEPT)
                  FAC(IMSPROD)
TSS ADDTO(OWNUSER) OTRAN(PAYINQ)
TSS ADDTO(OWNUSER) CIMS(DIS)
TSS PERMIT(TERM1) OTRAN(PAYINQ)
                  FAC(IMSPROD)
TSS PERMIT(TERM1) CIMS(DIS)
                  FAC(IMSPROD)
```

In this example, the TSSISMU4 utility is executed with PARM='USER,OWNUSER,IMSPROD' and the SMU input is:

```
)( TRANSACT PAYINQ
   TERMINAL TERM1
   TERMINAL TERM2
```

The utility generates the following PERMIT commands:

```
TSS ADDTO(OWNUSER) OTRAN(PAYINQ)
TSS PERMIT(TERM1) OTRAN(PAYINQ) FAC(IMSPROD)
TSS PERMIT(TERM2) OTRAN(PAYINQ) FAC(IMSPROD)
```

TSSISMU4 Authorization Requirements

There are two steps to the TSSISMU4 utility process.

- Execute the TSSCFILE utility to obtain the current resource ownership information for IMS transactions and commands. This step should be executed by the MSCA to ensure that all security information is obtained.
- Executes the TSSISMU4 utility. There are no authorization requirements for the TSISMU4 utility.

Sample TSSCFILE JCL

This sample JCL executes the TSSCFILE utility:

```
//    JOB …
//TSSCFILE   EXEC PGM=TSSCFILE
//OUT        DD  DSN=tsscfile.output,
//           DISP=(NEW,CATLG,DELETE),
//           UNIT=SYSDA,
//           VOL=SER=vvvvvv,
//           SPACE=(CYL,(10,10),RLSE),
//           DCB=(RECFM=FB,LRECL=300)
//PRINT      DD  SYSOUT=*
//IN         DD  *
TSS WHOOWNS OTRAN(*)
TSS WHOOWNS CIMS(*)
//
```

OUT

This DD statement specifies the output data set created by the TSSCFILE utility containing the output of the TSS WHOOWNS commands. This data set is used as input to the TSSISMU4 utility.

This must be a sequential data set with a record length of 300.

PRINT

This DD statement specifies the utility report data set created by the TSSCFILE utility.

IN

This DD statement specifies the data set containing the TSS CFILE input commands. These commands are required and must be entered exactly as shown.

Sample TSSISMU4 JCL

This sample JCL executes the TSSISMU4 conversion utility:

```
//    JOB …
//TSSISMU4   EXEC PGM=TSSISMU4,
//           PARM='USER|ATS,owner,imsfac,department'
//TSSCFILE   DD  DSN=tsscfile.input,DISP=SHR
//SMU        DD  DSN=smu.input,DISP=SHR
//TSSCMDS    DD  DSN=tsscmds.output,DISP=SHR
//
```

USER|ATS

Controls whether the output from the utility is for a user-based conversion process (USER) or a terminal-based conversion process (ATS).

Owner

Specifies the ACID used in the ADDTO statement giving ownership of IMS transactions or commands that are not already owned. If different owning ACIDs are desired for different IMS resources, a placeholder literal such as ???????? can be specified. The TSSCMDS data set can then be edited, and the correct values for the owning ACIDs substituted before the commands are executed.

Imsfac

Specifies the facility name of the IMS control region whose SMU input is being converted. This value is used in the TSS CREATE statements for the terminal ACIDS (ATS conversion only) and the PERMIT statements that are generated to give access to the transactions and commands in the IMS region.

Department

This parameter is only required if the first value in the parameter list is ATS. The value is used in the TSS CREATE statements to create ACIDs for the terminal names. The terminal ACIDs are defined in the specified department.

TSSCFILE

This DD statement specifies the input data set containing the TSSCFILE input for transaction and command ownership.

This must be a sequential data set with a record length of 300.

SMU

This DD statement specifies the input data set containing the SMU statements for the IMS region being processed.

This must be a sequential data set with a record length of 80.

TSSCMDS

This DD statement specifies the output data set created by the TSSISMU4 utility, containing the TSS statements generated by the conversion utility. This must be a sequential data set with a record length of 80.

How TSSISMU4 Processing Works

The TSSISMU4 conversion utility:

- Parses the USER|ATS, owner, facility, and department values from the PARM= execution parameter in the TSSISMU4 job stream
- Processes the information from the TSSCFILE input data set to create storage tables of transaction and command resource ownership.
- Processes the SMU information from the data set specified in the SMU DD statement in the execution JCL.
- Identifies the)(TERMINAL control statements and extracts the LTERM name
  - Processes the data statements that follow the)(TERMINAL control statement
  - Identifies any TRANSACT or COMMAND data statements and performs the conversion process for this terminal / transaction or terminal / command combination.
  - Generates a TSS CREATE command to create an ACID for the terminal if the utility control parameter is ATS and this is the first time the LTERM name has been processed

    The department for the ACID is the department specified in the PARM= parameter in the utility JCL, and the facility for the ACID is the facility in the PARM= parameter.
  - For both ATS and USER, checks the in-storage ownership tables to see if the transaction or command is currently owned in CA Top Secret
  - Generates an ADDTO statement to give resource ownership if the resource is not currently owned.

    The ACID in the ADDTO statement is the owner specified in the PARM= input parameter.
  - Generates a PERMIT statement giving the terminal ACID access to the transaction or command from the facility of the IMS region.

- Identifies)(TRANSACT control statements and extracts the transaction name from the control statement
  - Processes the data statements that follow the)(TRANSACT control statement.
  - Finds any TERMINAL data statements and performs the conversion process for this terminal / transaction combination.
  - Generates a TSS CREATE command to create an ACID for the terminal if the utility control parameter is ATS, and this is the first time the LTERM name has been processed.

    The department for the ACID is the department specified in the PARM= parameter in the utility JCL, and the facility for the ACID is the facility in the PARM= parameter.
  - Checks the in-storage ownership tables to see if the transaction is currently owned in CA Top Secret for both ATS and USER.
  - Generates an ADDTO statement to give resource ownership if the transaction is not currently owned.

    The ACID in the ADDTO statement is the owner specified in the PARM= input parameter.
  - Generates a PERMIT statement giving the terminal ACID access to the transaction from the facility of the IMS region.
- Identifies)(COMMAND control statements and extracts the command name from the control statement.
  - Processes the data statements that follow the)(COMMAND control statement.
  - Performs the conversion process for this terminal / command combination if a TERMINAL data statement is found.
  - Generates a TSS CREATE command to create an ACID for the terminal if the utility control parameter is ATS and this is the first time the LTERM name has been processed

    The department for the ACID is the department specified in the PARM= parameter in the utility JCL and the facility for the ACID is the facility in the PARM= parameter.
  - Checks the in-storage ownership tables to see if the command is currently owned in CA Top Secret for both ATS and USER.
  - Generates an ADDTO statement to give resource ownership if the command is not currently owned

    The ACID in the ADDTO statement is the owner specified in the PARM= input parameter.
  - Generates a PERMIT statement giving the terminal ACID access to the command from the facility of the IMS region.

Executing the TSS Commands

Review the TSS commands generated by the TSSISMU4 utility for accuracy before you execute them.

The following sample JCL executes the TSS statements in a batch TSO job.

```
//    JOB …
//IKJEFT01     EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT     DD   SYSOUT=*
//SYSPRINT     DD   SYSOUT=*
//SYSTSIN      DD   DSN=tsscmds.input,DISP=SHR
//
```

To execute the TSS statements, the user must have an ACID with sufficient authority to perform the CREATE, ADDTO, and PERMIT commands.

SMU Terminal Security to CA Top Secret Migration

To migrate from SMU terminal security to CA Top Secret:

- Determine whether you need to perform the conversion of the SMU terminal based security definitions. If existing CA Top Secret transaction and command security policy is sufficient, conversion of the SMU terminal security may not be necessary.
- Determine whether you want to convert the SMU terminal security definitions to a user-based CA Top Secret policy or a terminal-based (ATS) CA Top Secret policy.
- Determine the facility of the IMS region whose SMU security is being converted.
- Determine whether you want to assign resource ownership to a single ID, or use a substitution literal in the owner value of the PARM= execution parameters.
- If you are performing a terminal-based (ATS) conversion, determine the department in which the terminal ACIDs are created.
- Run the TSSCFILE utility to obtain the ownership information for IMS transactions and commands.
- Run the TSSISMU4 conversion utility program using the TSSCFILE input and the SMU statements for the IMS region being converted.
- Review the TSS command statements created by the TSSISMU4 utility for accuracy.
- If you are performing a user-based conversion, the utility generates a PERMIT statement for the IMS transactions and commands with a dummy value in the ACID initialized to the IMS logical terminal name. Determine who the users are who require access to the transaction or command, and change the ACID value in the PERMIT statement to reflect those users. If necessary, the PERMIT statement can be duplicated to permit access from different sets of users.
- Execute the TSS commands to update CA Top Secret.
- Change the SECURITY macro in the IMS system definition to turn off SMU terminal security, and to enable CA Top Secret terminal and command security.
- Run the IMS system definition.
- Change the IMS execution parameters to enable terminal and command security.
- Restart IMS

SMU Functionality without CA Top Secret Equivalent

The following functionality in SMU has no direct parallel in CA Top Secret processing:

- Requirements for terminal signon
- Passwords for IMS resources

The SMU signon requirements functionality is addressed in the IMS system definition (IMS stage 1). Passwords for IMS resources cannot be implemented in an IMS external security environment.

Use the TSSISMU5 utility to report the statements in a SMU security definition that cannot be converted to a CA Top Secret equivalent.

SMU Signon Requirements

SMU used the)(SIGN and STERM statements to define the IMS static terminals that must sign on before they can enter transactions and commands.

Beginning in IMS r9.1, IMS provides the same functionality with the SIGNON= initialization parameter in the IMS DFSDCxxx parameter list, and with the OPTIONS=SIGNON|NOSIGNON parameter on the TYPE or TERMINAL macro statements in the IMS system definition (IMS stage 1).

If you specify STERM ALL in your SMU signon security definition, specify SIGNON=ALL in your IMS DFSDCxxx initialization parameter list, and remove the SMU signon definitions.

If you list specific static terminals in your SMU signon security definition, add the OPTIONS=SIGNON parameter to the TYPE or TERMINAL macros for those terminals in your IMS system definition (IMS stage 1), and run your system definition process. Then specify SIGNON=SPECIFIC in your IMS DFSDCxxx initialization parameter list, and remove the SMU signon definitions.

IMS provides a conversion program that will process your SMU definitions and your IMS system definitions, and provides a system definition updated with OPTION=SIGNON specifications for those terminals identified in the SMU definitions as required to signon. For information, see your IMS documentation.

Password for IMS Resources

In the SMU security definitions, you can assign a password to several different kinds of IMS resources. Before a resource is used, the corresponding password must be provided. No equivalent functionality was added to IMS r9.1 to provide this capability. In CA Top Secret, only users have passwords; there is no mechanism to assign a password to a resource.

The TSSISMU5 utility

Use the TSSISMU5 utility to generate a report of the SMU security definitions that cannot be converted to equivalent CA Top Secret functionality. The utility processes the SMU security definitions for the IMS region being converted. For each SMU definition that does not have a direct equivalent in CA Top Secret, the utility generates output to a report. Use this report to ensure that equivalent functionality is provided by either IMS system definitions or some other mechanism.

The SMU security definitions that do not have an equivalent in CA Top Secret policy definitions are:

- Requirements for terminal signon
- Passwords for IMS resources

Sample TSSISMU5 JCL

This sample JCL executes the TSSISMU5 conversion utility.

```
//    JOB …
//TSSISMU5    EXEC PGM=TSSISMU5
//SMU         DD DSN=smu.input,DISP=SHR
//REPORT      DD SYSOUT=*
//
```

SMU

This DD statement specifies the input data set containing the SMU statements for the IMS region being processed.

This must be a sequential data set with a record length of 80.

REPORT

This DD statement specifies the output data set created by the TSSISMU5 utility, containing a report listing the SMU security definitions that cannot be converted to equivalent functionality in CA Top Secret.

If the DD statement specifies a data set rather than SYSOUT, the data set must sequential with a record length of 80.

How TSSISMU5 Processing Works

The TSSISMU5 conversion utility processes the SMU information from the data set specified in the SMU DD statement in the execution JCL.

When the utility:

- Identifies a)(SIGN control statement it generates report lines for the)(SIGN statement and any STERM data statements that follow it.
- Identifies a)(PASSWORD control statement it generates report lines for the)(PASSWORD statement and any data statements that follow it.
- Identifies a)(COMMAND,)(DATABASE,)(PROGRAM,)(PTERM,)(TERMINAL, or)(TRANSACT control statement, it saves the control statement.

  If the utility finds a PASSWORD data statement following the control statement it generates report lines for the control statement and the PASSWORD data statement that follows it.

Conversion Utilities Messages

Unable to locate SAF IVT

Reason:

A critical CA SAF control block could not be located. An improper installation of the security product or corruption of common storage may cause this. The utility terminates.

Action:

Contact CA Technical Support.

Module:

TSSISMU1, TSSISMU2, TSSISMU3, TSSISMU4, TSSISMU5

Error opening the SMU input file

Reason:

The utility encountered an error finding or opening the SMU input file. The utility terminates.

Action:

Make sure that the SMU DD statement exists and points to a sequential data set, with an LRECL of 80, containing the SMU security statements to be processed.

Module:

TSSISMU1, TSSISMU2, TSSISMU3, TSSISMU4, TSSISMU5

Error opening the TSSCMDS output file

Reason:

The utility encountered an error finding or opening the TSSCMDS output file. The utility terminates.

Action:

Make sure that the TSSCMDS DD statement exists and points to a sequential data set, with an LRECL of 80, which is used to contain the output commands from the utility.

Module:

TSSISMU1, TSSISMU2, TSSISMU3, TSSISMU4

Error opening the TSSCFILE input file

Reason:

The utility encountered an error finding or opening the TSSCFILE input file. The utility terminates.

Action:

Make sure that the TSSCFILE DD statement points to a sequential data set, with an LRECL of 300, containing the CFILE input data.

Module:

TSSISMU2, TSSISMU3, TSSISMU4

Error opening the REPORT output file

Reason:

The utility encountered an error opening the REPORT input file. The utility terminates.

Action:

Make sure that the REPORT DD statement points to a sequential data set, with an LRECL of 80.

Module:

TSSISMU1, TSSISMU3, TSSISMU5

Error opening the STG1IN input file

Reason:

The utility encountered an error finding or opening the STG1IN input file. The utility terminates.

Action:

Make sure that the STG1IN DD statement points to a sequential data set, with an LRECL of 80, containing the IMS system definition statements to be processed.

Module:

TSSISMU1, TSSISMU3

Error opening the STG1OUT output file

Reason:

The utility encountered an error finding or opening the STG1OUT output file. The utility terminates.

Action:

Make sure that the STG1OUT DD statement exists and points to a sequential data set, with an LRECL of 80, which will contain the updated IMS system definition statements.

Module:

TSSISMU1, TSSISMU3

Invalid value in the PARM='department,facility' input parameter

Invalid value in the PARM='msca, owner,facility' input parameter

Invalid value in the PARM=facility input parameter

Invalid value in the PARM='ATS|USER,owner,facility,dept' input parameter

Reason:

One of the values specified in the PARM= parameter is missing, or the value specified was longer than eight characters. The utility terminates.

Action:

Specify valid values in the PARM= parameter in the utility JCL.

Module:

TSSISMU1, TSSISMU2, TSSISMU3, TSSISMU4

Insufficient storage for execution

Reason:

The utility was unable to obtain sufficient storage for internal work areas. The utility terminates.

Action:

Increase the region available for the utility execution.

Module:

TSSISMU1, TSSISMU2, TSSISMU3, TSSISMU4

Invalid SMU input - record nn

Reason:

The utility encountered a syntax error processing a SMU input record. The relative record number for the SMU record is nn. The utility terminates.

Action:

Edit the SMU input and determine the syntax error in record number nn. Some of the reasons for this message are a command name that is longer than ten characters or a transaction name longer than eight characters. Correct the error.

Module:

TSSISMU1, TSSISMU2, TSSISMU3, TSSISMU4, TSSISMU5

There were no AOI transactions processed.

Reason:

This message can occur in the TSSISMU1 AOI=TRAN report or the TSSISMU3 AOI=YES report. The message is issued when the conversion utility did not find any valid CTRANS statements in the SMU security definitions. Since there are no transactions that issue AOI commands, no transaction definitions in the IMS system definition (stage 1) need to be modified.

Action:

This is an informational message. No action is required.

Module:

TSSISMU1, TSSISMU3

Index