CA TLMS® Tape Management

TLMS_Configuration_ENU

Release 12.6 Second Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA ACF2™ for z/OS (CA ACF2 for z/OS)
- CA Chorus Software Manager (CA CSM)
- CA Disk Backup and Restore (CA Disk)
- CA JCLCheck™ Workload Automation (CA JCLCheck WA)
- CA OPS/MVS® Event Management and Automation (CA OPS/MVS EMA)
- CA Service Desk (CA Service Desk)
- CA TLMS® Tape Management (CA TLMS)
- CA Top Secret® for z/OS (CA Top Secret for z/OS)
- CA Vtape™ Virtual Tape System (CA Vtape VTS)
- CA Workload Automation Restart Option for z/OS Schedulers (CA WA Restart Option for z/OS Schedulers)

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.

Contents

Chapter 4: User Exits and Macros 129

Chapter 5: Scratch Pooling 149

Chapter 1: System Components and Execution

This section contains the following topics:

About This Guide

This guide presents the technical information and procedures needed to configure and maintain CA TLMS. It is designed primarily for systems programmers to answer questions about configuration, maintenance, and troubleshooting. Detailed information on product installation is contained in the *Installation Guide*.

The utilities used by the systems programmer are also documented in this guide.

Major Processing Modules

The online processing performed by CA TLMS is contained within a collection of major modules. Some of these modules are briefly described in the following paragraphs.

TLMSMAIN

TLMSMAIN is the root module of CA TLMS. When attached as a subtask by the CTS, TLMSMAIN performs housekeeping functions, and then waits on the completion of one of the following three events:

- A transaction to process, as indicated by a cross-memory post from the TLMSQMGR module.
- A command to process, as indicated by a post from CTS.
- Completion of the time-out interval established to avoid an S522 abnormal termination.

The following programs are executed by TLMSMAIN:

TLMSMSGQ

This message queue processor is called by TLMSMAIN when a transaction has been queued for CA TLMS by the TLMSQMGR SVC. The queue processor invokes the TLMSQMGR SVC to dequeue and retrieve the transaction, then examines the transaction type to determine the appropriate processing path. If a gummed volume label is appropriate to the transaction in process, the label printing routine is invoked here. If the file access process has not concluded successfully, the operator is notified. Finally, if the initiator of the transaction is waiting for completion (for example, the TLMSOPEN routine at OPEN for output time), this module notifies it by post.

TLMSUPDT

TLMSUPDT manages access to the Volume Master File at the logical level. Records are retrieved from the VMF, updated with the data from transaction records passed from TLMSMSGQ, and written back to the file. If the optional ALOG file is indicated, a log record is written (for eventual use if recovery is necessary). File utility functions, such as OPEN and CLOSE, are managed by this module.

TLMSVMRW

Physical access to the Volume Master File is accomplished by routines in program TLMSVMRW, which is called by TLMSUPDT. TLMSVMRW performs the physical function (read, write and so forth) after ensuring serialization of the resource (VMF) using an ENQUEUE/RESERVE technique. When this program is called, the target record is defined by key, such as VSN; this program contains the detailed code necessary to locate the record through the use of control/index arrays and so forth.

Note: For more information about the CTS address space and the subtasks it supports, see the chapter "Common Tape System (see page 75)" in this guide.

TLMSOSMM

This module is invoked by CA TLMS operating system intercepts used for processing during OPEN/CLOSE/EOV. All functions are consolidated into this module, which is automatically installed by CAIRIM (CA Resource Initialization Manager component). The following programs are executed by the operating system intercepts from TLMSOSMM.

- TLMSCLSE

  TLMSCLSE is a module executed as an exit from CLOSE. Linked into the common SVC, it is activated by an SVC called from the intercept. CLOSE modules differ depending on the operating system in use.

  TLMSCLSE examines the CLOSE in process and builds a CA TLMS transaction with a type code of 1 for output tape or 2 for input tape. For output files, a type 1 transaction may cause a gummed label to be printed, and the DSN and run statistics to be recorded in the VMF.

  TLMSCLSE passes the transaction to CA TLMS by issuing an SVC to the TLMSQMGR module. TLMSCLSE does not wait for the process to complete; an immediate return to CLOSE processing takes place.

- TLMSOPEN

  TLMSOPEN is a module executed as an exit from OPEN. Linked into the common SVC, it is activated by an SVC called from the intercept. The OPEN modules differ depending on the Operating System in use.

  TLMSOPEN examines the OPEN in process and builds a CA TLMS transaction with a type code of 6. For output files, a type 6 transaction is intended to obtain permission to write upon the mounted volume, and to flag the VMF that the volume is in use. For input files, a type 6 transaction is used to verify the input DSN against the VMF DSN, if the IDSNVER system option is set to YES.

  TLMSOPEN passes the transaction to CA TLMS by issuing an SVC to the TLMSQMGR module. OPEN then waits for the process completion post. When posted, a return vector is passed back to OPEN, reflecting the status of the transaction: good, use tape; or bad, demount tape and try again.

- TLMSQMGR

  TLMSQMGR is linked into the common SVC. For CA TLMS, TLMSQMGR performs all cross-memory services when invoked by either of the exit modules (SVCs): TLMSOPEN, TLMSCLSE. These exit modules build a CA TLMS transaction, then invoke TLMSQMGR to queue the transaction (in a CSA queue), and notify TLMSMAIN that there is a transaction to be processed.

  At a later point in the cycle, TLMSQMGR is again invoked, by TLMSMSGQ. TLMSQMGR dequeues the transaction, and presents it to TLMSMSGQ.

- TLMSSECU

  The module accessed through SVC which manages security calls to the operating system security component.

BATCH

For purposes of this discussion, the periodic process of interrogating the Volume and Retention Master files, determining and reporting retention, movement and scratch activities, and updates to the VMF reflecting these activities, are lumped together under the term BATCH. During BATCH processing, the VMF is read directly; however, updates are effected through the same mechanism (SVCs) as the online updates.

CA TLMS Process Flow

The following pages describe the major process flow paths within CA TLMS. Paths which correspond to transactions are identified. The process flows are:

- Output Tape OPEN (transaction code 6)
- Input Tape OPEN (transaction code 6)
- Output Tape CLOSE (transaction code 1)
- Input Tape CLOSE (transaction code 2)
- TSO-Originated Inquiry and Update (transaction code 4)
- Console-Originated Inquiry and Update (transaction code 3)
- Tape Retention (Batch) System File Access and Update (transaction code 5)
- Batch Inquiry/Update (transaction code 4)
- Cross Memory get Request (transaction code 9)

The following notes apply to the process flows discussed in this section:

- Only *steady state* processes are described. Initialization processes are not described, but are assumed to have been successfully concluded.
- Specific references to operating system intercept locations (that is, module names) have been avoided, since they vary depending on which operating system version is employed.
- Specific details of implementation have been avoided except where necessary to clarify the process flow.

Output Tape OPEN Process Steps

1. An SVC call, implemented using an operating system intercept, is issued out of OPEN to perform scratch verification and data set protection.
2. The TLMSOPEN module gains control, builds a CA TLMS type 6 transaction, and issues a TLMSQMGR SVC.
3. The TLMSQMGR module gains control, receives the transaction, and cross-memory posts TLMSMAIN.
4. TLMSMAIN is activated by the cross-memory post.
5. TLMSMAIN responds to the post by calling TLMSMSGQ.
6. TLMSMSGQ receives a copy of the transaction by calling the TLMSQMGR SVC and then calls TLMSUPDT for processing.
7. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
8. The VMF is accessed by volume serial number (VSN).
9. If the retrieved VMF record indicates that the volume is in scratch status, the record is updated to indicate that the volume is *in use*, and is written back to the VMF.
10. (Optional) A copy of the transaction is written to the Alternate Log file if the RECOVRY system option is set to ALTLOG.
11. TLMSMSGQ is notified that the transaction is successful (the use of this volume may proceed), or that it is unsuccessful (another volume must be selected).
12. In either case, the initiating task is posted to report the status of the transaction. In this case, OPEN is waiting for this post.
13. OPEN is activated by the post from TLMSMSGQ.
14. Using the return vector to OPEN, OPEN either proceeds or demounts the volume and tries again.

Input Tape OPEN Process Steps

1. An SVC call, implemented using an operating system intercept, is issued out of OPEN to perform input data set name verification.
2. The TLMSOPEN module gains control, builds a CA TLMS type 6 transaction, and issues the TLMSQMGR SVC.
3. The TLMSQMGR gains control, receives the transaction and cross-memory posts TLMSMAIN.
4. TLMSMAIN is activated by the cross-memory post.
5. TLMSMAIN responds to the post by calling TLMSMSGQ.
6. TLMSMSGQ receives a copy of the transaction through an SVC call to TLMSQMGR. It then calls TLMSUPDT for processing.

7. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
8. The VMF is accessed by volume serial number (VSN).
9. (Optional) The retrieved VMF record is examined to validate the DSN being opened if the IDSNVER system option is set to YES.
10. TLMSMSGQ is notified that the transaction is successful (this is the same DSN), or that it is unsuccessful (this is not the same DSN).
11. The initiating task is posted to report the status of the transaction. In this case, OPEN is posted.
12. OPEN is activated by the post from TLMSMSGQ.
13. Using the return vector to OPEN, OPEN either proceeds or dismounts the volume and tries again.

Output Tape CLOSE Process Steps

1. An SVC call, implemented using an operating system intercept, is issued out of CLOSE.
2. The TLMSCLSE module gains control, builds a CA TLMS type 1 transaction, and issues the TLMSQMGR SVC (without wait).
3. The TLMSQMGR gains control, receives the transaction, and cross-memory posts TLMSMAIN.
4. TLMSMAIN is activated by the cross-memory post.
5. TLMSMAIN responds to the post by calling TLMSMSGQ.
6. TLMSMSGQ receives a copy of the transaction through an SVC call to TLMSQMGR. It then calls the label processing routines in TLMSVMRW.
7. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
8. The VMF is accessed by volume serial number (VSN).
9. The retrieved VMF record is updated with the new data set name and with run statistics.
10. (Optional) A copy of the transaction is written to the Alternate Log file if the RECOVRY system option is set to ALTLOG.
11. TLMSMSGQ is notified that the update is complete.

Input Tape CLOSE Process Steps

1. An SVC call, implemented using an operating system intercept, is issued out of CLOSE.
2. The TLMSCLSE module gains control, builds a type 2 transaction, and issues the TLMSQMGR SVC (without wait).
3. The TLMSQMGR gains control, receives the transaction, and cross-memory posts TLMSMAIN.
4. TLMSMAIN is activated by the cross-memory post.
5. TLMSMAIN responds to the post by calling TLMSMSGQ.
6. TLMSMSGQ receives a copy of the transaction through an SVC call to TLMSQMGR, then, it calls the label processing routines in TLMSVMRW.
7. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
8. The VMF is accessed by volume serial number (VSN).
9. The retrieved VMF record is updated with run statistics.
10. (Optional) A copy of the transaction is written to the Alternate Log file if the RECOVRY system option is set to ALTLOG.
11. TLMSMSGQ is notified that the update is complete.

TSO Inquiry With Update Process Steps

1. A VMF inquiry/update is initiated by a TSO user.
2. TSO accesses the VMF in read-only mode using a private copy of the TLMSVMRW module.
3. TLMSVMRW retrieves a record from the VMF using the VSN (retrieval by DSN is also possible, using a VMF Index file, not shown in this discussion).
4. To initiate an update, a CA TLMS transaction with a transaction code of 3 is built, and the TLMSQMGR SVC is issued (no wait).
5. The TLMSQMGR module gains control, receives the transaction from the TSO module, and cross-memory posts TLMSMAIN.
6. TLMSMAIN is activated by the cross-memory post.
7. TLMSMAIN responds to the post by calling TLMSMSGQ.

8. TLMSMSGQ receives a copy of the transaction by issuing the TLMSQMGR SVC, then calls TLMSUPDT.
9. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
10. The VMF is accessed by VSN.
11. The retrieved VMF record is updated.
12. (Optional) A copy of the transaction is written to the Alternate Log file if the RECOVRY system option is set to ALTLOG.
13. TLMSMSGQ is notified that the update is completed.

Console Inquiry With Update Process Steps

1. A VMF inquiry/update through reply to a WTOR from TLMSINQR places a transaction in the queue through TLMSMSGQ. TLMSINQR can be started as a subtask of CTS, a z/OS started task, or as a batch job.
2. TLMSMSGQ receives the transaction and calls TLMSUPDT.
3. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
4. The VMF is accessed by volume serial number (VSN).
5. The retrieved VMF record is updated.
6. (Optional) A copy of the transaction is written to the Alternate Log file if the RECOVRY system option is set to ALTLOG.
7. TLMSMSGQ is notified that the update is completed.

Batch Tape Retention System Process Steps

1. The Tape Retention System (TRS) runs in batch mode.
2. TRS accesses the VMF in read-only mode using its own private copy of TLMSVMRW.
3. The Retention Master File (RMF) is accessed to determine any special retention handling to be applied on a DSN basis.
4. Various printed reports are prepared.
5. VMF updates (for example, scratches) are performed by building CA TLMS transaction, type 5, and writing it to a transaction file. TLMSTRAN reads this file after a successful TRS run and issues the TLMSQMGR SVC.

6. TLMSQMGR is activated through the SVC.
7. The TLMSQMGR module gains control, receives each transaction, and cross-memory posts TLMSMAIN.
8. TLMSMAIN is activated by the cross-memory post.
9. TLMSMAIN responds to the post by calling TLMSMSGQ.
10. TLMSMSGQ receives a copy of the transaction by calling the TLMSQMGR SVC and then calls TLMSUPDT.
11. TLMSUPDT accesses the Volume Master File specified in the CTS procedure through routines in its copy of TLMSVMRW.
12. The VMF is accessed by volume serial number (VSN).
13. The retrieved VMF record is updated.
14. (Optional) A copy of the transaction is written to the Alternate Log file if the RECOVRY system option is set to ALTLOG.
15. TLMSMSGQ is notified that the update is completed.

Operating System Intercepts

CA TLMS Operating System Intercepts (OSIs) intercept processing between the appropriate IBM modules to enable various data set protection functions for standard label (SL), ANSI label (AL), nonlabeled (NL), and bypass label (BLP) tapes. As needed during the intercepts, the CA TLMS SVC (TLMSOSMM) is invoked so that data can be captured from IBM control blocks and sent to CA TLMS through a transaction queue. OMODVOL1 and EMODVOL1 are intercepted and processed according to CA TLMS options. SMS tape user exits are also intercepted.

These OSIs work for all z/OS operating systems. They are not dependent on instruction changes within IBM modules and are not affected by IBM maintenance to OPEN/CLOSE/EOV modules. The OSIs are always dynamically applied and may be dynamically removed.

When a tape is mounted for output, CA TLMS does a verification of each OSI and immediately reports the removal of any OSIs by other products or processes.

NO SMP/E or selection process is required. The intercepts are automatically installed when TLMSRIM is run (normally with CAS9 after an IPL) and verified each time CA TLMS is started. If needed, the OSIs can be dynamically removed by running TLMSRIM with the REINIT parameter.

The following is an example of a CAS9 parameter to apply OSIs:

```
INIT
```

The following is an example of a CAS9 parameter to remove OSIs:

```
REINIT,OSI=NO
```

CA TLMS Major Features

The following sections explain the major features of TLMS.

Auxiliary Messages

One of the designations required during Volume Master File initialization is MESSAGE=Y|N. If MESSAGE=Y is specified, you can use auxiliary messages to display special handling instructions for data sets. Auxiliary messages can provide routing and disposition instructions such as:

- Holding a tape data set in the data center for an upcoming job
- Delivering a tape data set to an outside company
- Calling a person when the job ends
- Immediately moving a tape data set to a certain area
- Returning a tape data set to the library only under a specific condition

These messages are automatically sent to the operator console when a data set is closed. They can optionally be routed to a different console by specifying that console number using the ROUTAUX= system option.

There are two ways to define auxiliary messages for data sets:

- When a data set is created, use the EXPDT JCL parameter and specify LABEL=EXPDT=970*mm*, where *mm* is the number of the message that is to be displayed at the close of the data set.
- Use the AUX= system option to associate specific data set names or qualifiers with specific message numbers. This allows you to display the same message for all data sets that begin with the same qualifier.

Auxiliary messages can be updated and displayed either online or in batch mode, as detailed in the *User Guide*.

Foreign Volume Processing

Foreign volumes are defined as any that reside outside the volume serial number range of the Volume Master File, and for which no data set or volume serial number information was recorded (other than an event audit transaction). Volume serial numbers outside the range of the VMF are automatically processed as foreign volumes.

You will occasionally need to manage foreign volumes having volume serial numbers which are within a range defined in the Volume Master File for CA TLMS controlled tapes. You will need to direct CA TLMS to specifically ignore these volumes, that is, not to update the duplicate volume serial number in the Volume Master File.

To force CA TLMS processing as a foreign tape volume, specify one of the following on the DD statement for the data set

- LABEL=EXPDT=98000
- SPACE=(1,(1,1))

When foreign volume processing is selected through the use of JCL keywords, any data set may be read for input after passing system security requirements. The same process applies to output; however, a WTOR is issued to confirm or deny the write, providing an additional level of file protection. The CMSG option can be used to suppress the message and bypass operator intervention, if desired.

Two system options in member TLMSIPO of CAI.CTAPOPTN, FORSPEC and FORSEC, can be used to extend the control for foreign volumes and bypass volume functions. FORSPEC determines whether a foreign volume can be written on a specific volume request (requires VOL=SER= in JCL), or used for nonspecific (scratch) requests. FORSEC determines whether security checking is activated for foreign volume processing in JCL (LABEL=EXPDT=98000) when the security option SECURE is set to YES. When CA TLMS security is activated, access to foreign volumes can be controlled by defining FORRES (volume within VMF range) and FORNORES (volume outside VMF range) to the operating system security component.

The data set information will be printed on an external gummed label for an output data set, but it is not recorded in the Volume Master File. CA TLMS logs the transaction to the ALOG or SMF. You can receive both a summary and a detailed report of foreign tape activity by requesting report TLERPT19, System Activity Analysis. (See the "Reports" chapter in the *User Guide*.)

Extended System Failure Protection

CA TLMS automatically provides extended tape data set protection against catastrophic system failure for all tape data sets at OPEN for output time:

- CA TLMS changes the scratch status to NONSCRATCH and the data set name to TLMSII-CRASH-PROTECTED in the Volume Master File at OPEN for output time.
- The data set name is changed to the actual data set name when the data set is closed and normal CA TLMS updates are made. By updating in this sequence, volumes mounted for output data sets are marked NONSCRATCH when processing starts. If an operating system failure occurs during the processing of the jobs, the volumes currently in use are protected against reuse when the operating system is restarted. The same job (job name) will be allowed to use the volumes for output without changing the scratch status of the volume.
- The special identity inserted in the data set name aids in identifying volumes in use when a system failure occurs. These volumes must be manually changed to SCRATCH status before they can be reused for output by another job.

If you do not want extended protection for all tape data sets, specify the system option PROTECT=SELECT; this restricts protection to selected data sets only. Then, select the data sets to be "crash-protected" by coding SPACE=(1,(1,2)) on their DD statements.

Note: Crash protection applies only to file 1 on a multi-data set volume or set of volumes.

Note: The special identifier remains in the data set name field if no 'CLOSE' event is intercepted. This may occur for reasons other than a system crash.

Input Data Set Name Verification

The optional input data set name verification feature compares the fully qualified data set name and file sequence number furnished in the JCL for all input files against the file numbers and data set names recorded in the Volume Master File. Extending data set name validation from the system default of 17 characters to the entire 44-character data set name enhances data set security.

- If a match is found, the tape is accepted and processing continues normally.
- If a match is not found, the tape is rejected, and the following message is sent to the console:

  CAT9030I volser ON DRIVE dddd INPUT DSN INVAL RETURN TO LIBRARY

For SL tapes, the job is abended with an S713-04 completion code. If the tape label does not match the last 17 characters of the DSN field of the JCL, the job will abend with an S813 abend. For NL, NSL, and BLP tapes, the job will continue to mount and reject the same volume until canceled by the operator.

For BLP tapes, the file sequence number provided in the JCL is translated to the equivalent SL file number before the file number and data set name comparisons are made. If they do not match, the tape is rejected. For non-specific SL requests, the tape is rejected and a new mount requested. This prevents unauthorized persons from dumping header and trailer labels.

This feature does not apply to foreign tape volumes.

The input data set name verification feature is installed by setting the IDSNVER system option to YES.

External Data Manager (EDM) Processing

CA TLMS provides EDM support by assigning volume ownership to the EDM and giving it full control of the contents. The EDM is identified by matching information in CTOEDM00 (the owner table). There may be multiple EDMs, each having a separate EDM ID, and multiple sets of definitions for each EDM ID. CA TLMS applies volume ownership by placing the EDM ID in the OWNER field of the volume base record.

This option relates DSN, job name and program name criteria to an EDM ID. CA TLMS assigns the EDM ID to each transaction with matching criteria. The transaction is said to be "from an EDM."

CA TLMS processes EDM (or owned) tapes as follows:

- Each tape created from an EDM is recorded in the VMF as volume 1 of 1, with the DSN of the first data set on that tape. This is done regardless of the number of tapes and/or data sets actually related to it.

- CA TLMS will ensure that only files from matching EDM IDs are written on a tape and that files from EDMs are not written on non-EDM tapes.
- The EDM is responsible for the contents of its tapes. CA TLMS will allow the EDM to write or overwrite any tape it owns and has specifically requested. This permits the EDM to reuse a tape without scratching it.
- Only the EDM can scratch a tape it owns. Therefore, each EDM is responsible for scratching tapes it no longer needs. This may be done using DISP=(OLD,DELETE) or a manual transaction to scratch.

You may use the Tape Retention System (TRS) to control movement of EDM tapes. No special coding is required, because EDM ownership is recorded in the VMF. EDM tapes may be processed separately or in a group with other tapes. Since only the EDM can scratch its tapes, it is a good practice to use type 7 (manual retention) for the last location of EDM tapes. TRS will not scratch the tapes no matter what RMF rules are coded, but it will issue error messages showing the tape was retained for EDM.

Tape pools defined for EDMs using the CTOEDM00 member are pointed to by the EDM= option in TLMSIPO.

Dynamic VMF Extend

CA TLMS lets you switch to a new VMF while still processing tapes. The new VMF must be large enough to contain all the active records that are in the old VMF. This new VMF can reside on the same volume or a different volume than the original VMF.

CATVMFX controls the dynamic VMF extend process. During the extend process, all TLMSs add a new VMF and perform synchronous updates to both VMFs while records from the old VMF are copied to the new VMF. After all active records are copied, CATVMFX creates a VMF alias pointing to the new VMF and all batch and online tape activity references the new VMF using this alias. After all jobs referencing the old VMF during the extend process are complete, all TLMSs will drop the old VMF, marking the end of this process.

Note: The old and new VMFs are updated synchronously during the extend process. You do not need to stop batch and online-tape activity. All jobs, referencing either the old or the new VMF, can run until finished.

Distributed Tape Support

CA TLMS lets you manage tapes created by selected backup products on distributed platforms. This support is provided through integration with free downloadable components known as iGateways and iSponsors. iGateways and iSponsors were developed by CA to allow a wide variety of distributed systems products to provide data to the BrightStor Portal product. CA TLMS uses the iSponsor/iGateway technology to extract media information from the product catalogs of the following products:

- CA ARCserve Backup for Windows
- CA ARCserve Backup for UNIX
- Veritas NetBackup for Windows
- Tivoli Storage Manager for Windows

Tapes created by these products can be tracked in the CA TLMS Volume Master File (VMF). The volume information in the VMF is a copy of the tape volume information extracted from the backup product database. No update to the backup product database is performed.

The Distributed Tape Support feature is managed by the DTS subtask of CTS. The DTS subtask is configured to communicate a list of servers defined in the CTSDTS00 control statement file. Communication with the iSponsor for the targeted backup product can be done using a START command or can be scheduled by defining schedules through the SCHD subtask of CTS. For more information about this feature, see DTS and SCHD subtasks in Common Tape System.

More Information:

CTS Scheduler (SCHD) (see page 119)

More Information:

Distributed Tape Support (DTS) (see page 112)
CTS Scheduler (SCHD) (see page 119)

Defining and Changing Installation Options

When TLMSRIM is run, CAI.CTAPOPTN contains the members that specify the installation options. Four members are required.

- CTOSCR*xx*
- CTONSM*xx*
- CTOEDM*xx*
- TLMSIPO

Instructions for control statements on the CAI.CTAPOPTN members are documented in this guide.

CTOSCRxx

CTOSCR*xx* specifies scratch pool definitions by naming the scratch tape pools and describing the ranges of the tapes belonging to the tape pool. The physical characteristics of the tapes for a specific scratch pool are user-defined. The contents of this member are defined through control statements processed by TLC6INIT.

Note: CTOSCR*xx* must be used if CTONSM*xx* is used.

Note: If you intend to specify scratch subpools only using the VOL=SER= parameter, you must still have at least one entry in CTONSMxx.

If specified, the table created from CTOSCR*xx* is used at OPEN and EOV for tapes mounted by nonspecific requests to verify that the mounted tape is from the requested scratch tape pool. The volume serial number of the tape must be in the scratch tape pool assigned. If any tape is mounted from outside the scratch pool range, CA TLMS calls for another tape and issues a console message. The tape is then demounted. Similarly, if a tape mount is satisfied by a pool-controlled volume but is not valid for the requesting data set name or job name, the volume is demounted with a SCRATCH.

CTONSMxx

CTONSMxx specifies data to assign a scratch tape pool by the following:

- SMS Management Class
- Data set name or job name
- Unit or EXPDT/RETPD

Control statements processed by TLMSRIM are defined in CTONSMxx. The table created from CTONSMxx is checked for the scratch-tape pool. The tapes mounted with nonspecific requests at OPEN and EOV, must belong to the scratch tape pool to be valid.

Note: Use CTONSMxx if using CTOSCRxx.

CTOEDMxx

CTOEDMxx contains the rules for identifying a tape as controlled by an External Data Manager (EDM). TLC6INIT loads the rules into an internal table and sorts the rules in ascending sequence (most to least specific) by data set name, program name, job name and dd name. The sorted table is loaded into common storage. You can define multiple rules with pattern-matching capabilities and exits that you can use to determine if a volume is EDM controlled.

Chapter 2: System Options

This section contains the following topics:

How This Chapter is Organized

This chapter provides detailed explanations of the system options library members and the options contained in each.

The CA TLMS installation options are maintained in various members of the CA common options library, CAI.CTAPOPTN. Most options are in the form of a switch indicating a YES or NO condition. Some are a variable series of entries provided by you to indicate special processing requirements.

The installation option members that reside in CAI.CTAPOPTN are listed below with a brief description of the types of options contained in each.

TLMSIPO

This member supplies general information, formatting and limit options, processing options, points to EDM, NSM, and SCR members, and real-time security options. It is also used to activate CA TLMS user exits.

CTOEDM00

This member is used to specify the external data manager parameter under the Common Tape System (CTS).

CTONSM00

This member is used to specify volume pools for Non-Specific Mounts (NSM).

CTOSCR00

This member is used to specify volume pools for scratch pools.

If making any changes to your TLMSIPO options, you can execute CAS9 with a parm of REINIT specified in parm for TLMS in CARIMPRM.

If an option is left unchanged (either during or after installation), the default parameter value shown for each option in the appropriate member and described in this chapter indicates the setting that will be assumed for operations.

CA TLMS installation options are presented in this chapter in alphabetical order within each of the topics listed below.

System Options (Member TLMSIPO)

Member TLMSIPO contains installation options that affect base CA TLMS operations. This member supplies default parameter values for the system options it contains. The CA TLMS CAIRIM initialization program TLMSRIM establishes a pointer to TLMSIPO in common storage so that all processing programs can gain access to the options.

The options contained in this member are listed in alphabetical order. A page reference is provided to assist you in quickly locating the detailed discussion in this chapter.

MANUAL

Manual update of Retention Schedules (see page 56)

NITEXIT

Control the Exit (TLMSXNIT) (see page 56)

NLSEC

No Label (NL) security processing (see page 57)

NOTLMS

CA TLMS NOT ACTIVE processing (see page 57)

NSLSEC

Nonstandard Label (NSL) security processing (see page 57)

NSM

Nonspecific mount suffix (see page 58)

OPNEXIT

OPEN exit (TLMSXOPN) (see page 58)

PAGESIZ

Number of lines per printer report page (see page 59)

PROMPT

User logon ID prompt (see page 59)

PROTECT

Restrict crash protection to selected data sets (see page 60)

QSIZE

Transaction queue entry limit (see page 60)

RECOVRY

Create backup records (see page 61)

ROUTAUX

Auxiliary message processing (see page 62)

ROUTINQ

Inquiry console processing (see page 63)

SCR

Scratch pool suffix (see page 64)

SCREXIT

Control the Scratch Exit (TLMSRACF) (see page 64)

SCRATCH

Volume scratch control (see page 65)

SECCLS

Security checking for DISP=YES or DISP=OUTPUT (see page 66)

SECEXIT

Security exit (TLMSXSEC) (see page 66)

SECINQ

Security checking for online inquiry (see page 67)

SECOPN

Security checking for open tape processing (see page 67)

SECURE

Global security option (see page 68)

SERIND

Use of out-of-service indicated tapes (see page 68)

SMS

SMS interface active (see page 69)

SPLEXIT

Control the Command Exit (CTSUX1G) (see page 69)

TRSEXIT

Tape retention exit (TLMSXTRS) (see page 70)

UNCATLG

Uncatalog action when scratch (see page 70)

UPDEXIT

Volume Master File update exit (TLMSXUPD) (see page 71)

USER

Internal User Security (see page 72)

VMFINAM

Data set name of the Volume Master File Index (see page 73)

VMFNAM

Data set name of the Volume Master File (see page 73)

VSNPAD

Volume serial numbers of less than six digits (see page 74)

VSNREQD

Volume serial number required at OPEN (see page 74)

Changing System Options

Should it be necessary to change any system option(s) after product installation, edit and apply changes to member TLMSIPO in CAI.CTAPOPTN, and execute the CAS9 procedure, with the CAIRIM parameter for CA TLMS set to PARM(REINIT). The following are valid PARMS for CA TLMS and CAS9:

INIT

Completes initialization. This is immediately after an IPL.

REINIT

Reinitialize the CA TLMS.

To display the current TLMSIPO options loaded, execute member CATOPTS from the CAI.CTAPPROC. Use the PRINT PARM in the execute statement.

Comment statements may be included by coding * in position 1 or /* before and */ after the comment. All parameters and comments must be completed before position 72. Position 72 must be blank.

These options may be validated for errors prior to the REINIT using program TLMSLDIP.

ABEND System Option—Scratch Abended Tapes

Use the ABEND system option to determine whether an abended tape can be scratched when TRS is executed and, if so, how many hours must elapse before it can be scratched.

This system option has the following format:

ABEND=[24|*hh*|NO|99]

hh

Specifies the number of hours that must elapse before an abended tape can be scratched.

Default: 24

NO

Specifies that abended tapes will be scratched and will be processed by the Tape Retention System as a normal data set. ABEND=99 is synonymous with ABEND=NO and the option displays ABEND=99 if either is specified.

ALTCTR System Option—Alternate Data Center ID

Use the ALTCTR system option to provide an alternate identification for the data center. An alternate data center ID can be used to move a nonscratch tape back to the primary data center without the tape retention system (TRS) resetting it to scratch status.

This system option has the following format:

ALTCTR=*xx*

xx

Specifies the alternate name for the data center.

Valid values: Any alphanumeric character

Default: D1

Example: ALTCTR System Option

ALTCTR=D2

AUX System Option—Data Set Name and Message Number for Auxiliary Messages

Use the AUX system option to generate specific auxiliary messages for data sets that require special handling instructions. This option dynamically builds table TLMSMTAB. If this option is set, the ROUTAUX option must be set to *YES*.

A maximum of 128 AUX= statements may be supplied.

The parameters specify the message number and starting position associated with the indicated data set name.

This system option has the following format:

AUX=*dsname* [,POS=*nn*] ,NMSG=*nn*

dsname

Specifies the data set name or prefix for auxiliary messages.

POS=*nn*

(Optional) Specifies the starting position in the data set name.

Default: 01

NMSG=*nn*

Specifies the message number associated with this data set.

Examples: AUX System Option

```
AUX=MSG.AAA,POS=7,NMSG=1
AUX=MSG.BBB,POS=7,NMSG=2
```

BLPSEC System Option—Bypass Label Processing (BLP) Security

Use the BLPSEC system option to specify security checking for Bypass Label Processing (BLP) when the SECURE option is set to YES.

This system option has the following format:

BLPSEC=[YES|NO]

NO

(Default) Specifies that security checking is not activated for Bypass Label Processing.

YES

Specifies that security checking is activated for Bypass Label Processing.

BRKCHN System Option—Break Volume/Data Set Chains

Use the BRKCHN system option to determine when chains are broken. The volume sequence, volume count, file sequence and file count fields in the VMF are reset. All data set-related fields are cleared.

This system option has the following format:

BRKCHN=[OPEN|CLOSE|SCRATCH]

OPEN

(Default) Specifies that a chain is broken when one of the volumes in the original chain is opened for output. Each volume within the chain becomes a single volume and all fields (except tape history) in the associated VMF records are reset.

CLOSE

Specifies that chains are broken when the first data set is closed on a volume.

Note: This option is supplied for compatibility with earlier releases. We recommend that this option be used in special circumstances only. The chains are broken at OPEN, regardless of this option being specified.

SCRATCH

Specifies that all chains are broken and previous information (except tape history) is reset. If a tape is scratched in error, the information and chaining sequence is erased; therefore, exercise caution when considering this selection.

CATDAYS System Option—Creation Date Retention Catalog Check

Use the CATDAYS system option to specify the number of days after creation before the z/OS catalog is checked for retention by TRS for catalog controlled volumes.

This system option has the following format:

CATDAYS=*n*

n

Specifies the number of days after creation that the z/OS catalog is checked for retention by TRS for cataloged controlled volumes.

Default: 1

Important! Specifying 0 (zero) may result in premature scratching of tapes during Tape Retention System execution if the data set has been closed, but not yet cataloged.

CDS System Option—Control Data Set Specifications

Use the CDS system option to define the controlling data set (CDS) for a multi-data set volume or set of volumes. This option dynamically builds user table TLMSCTAB.

The parameters specify the controlling data set name or prefix and its starting position. A maximum of 128 CDS= statements may be supplied.

This system option has the following format:

CDS=*dsname* POS=*nn*

dsname

Specifies the data set name or prefix for the controlling data set.

nn

Specifies the starting position in the data set name.

Default: 01

Example: CDS System Option

```
CDS=CDS.TEST,POS=7
CDS=CDS.PROD
```

CLSEXIT System Option—Control CLOSE/EOV Exit (TLMSXCLS)

Use the CLSEXIT option to determine whether the CA TLMS CLOSE/EOV exit (TLMSXCLS) is active or inactive.

Note: For more information about this exit, see the section User Exits and Macros (see page 129).

This system option has the following format:

CLSEXIT=[NO|*exitname*|YES]

NO

(Default) Specifies that the CA TLMS CLOSE/EOV exit is not activated.

exitname

The customer specifies the name for the exit.

YES

Specifies that the CA TLMS CLOSE/EOV exit is activated. The exit name is TLMSXCLS.

Example: CLSEXIT System Option

```
CLSEXIT=YES
```

CMDEXIT System Option—Control the Command Exit (TLMSXCMD)

Use the CMDEXIT system option to determine whether the CA TLMS command exit (TLMSXCMD) is active or inactive.

This system option has the following format:

CMDEXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CA TLMS command exit is not activated.

YES

Specifies that the CA TLMS COMMAND exit is activated. The exit name is TLMSXCMD

exitname

Specifies the customer name for the exit.

Example: CMDEXIT System Option

```
CMDEXIT=YES
```

More Information:

TLMSXCMD - Command Exit (see page 133)

CMSG System Option—Console Messages Processing

Use the CMSG system option to selectively suppress, delete, or highlight (nondeleteable) CA TLMS messages.

The parameters provide the criteria for displaying console messages.

A maximum of 200 CMSG= statements may be supplied. The following messages cannot be modified:

- Open processing—AT904**
- Close processing—CAT97***
- Queue processing—CADO9***
- Security processing—CATS****

This system option has the following format:

CMSG=*nnnnnn*, MSG=[DEL|SUP|NDEL]

CMSG=nnnnnn

Specifies the message or message prefix for action.

MSG=[DEL|SUP|NDEL]

Specifies one of the following:

NDEL

Non-deleteable message.

SUP

Suppressed message.

DEL

(Default) Deleteable message.

Example: CMSG System Option

```
CMSG=CAT4501P,MSG=SUP
```

COMPANY System Option—Company Name for Printed Reports

Use the COMPANY system option to specify the company name as you want it to appear on your reports.

This system option has the following format:

COMPANY='*company name*'

company name

Specifies the company name that is to appear on reports. Enclose in single quotes if name contains blanks or commas.

Range: Up to 30 characters

Example: COMPANY System Option

```
COMPANY='THE COMPANY'
```

DATACTR System Option—Primary Data Center ID

Use this system option to provide the primary identification for your data center.

This system option has the following format:

DATACTR=*xx*

xx

Specifies the data center. This must match the location ID of the data center specified in the RMF.

Valid values: Any two alphanumeric characters

Default: DC

Example: DATACTR System Option

```
DATACTR=0P
```

DATEFMT System Option—Preferred Date Format

Use the DATEFMT system option to establish your preferred date format pattern.

The value provided by this option can be overridden during execution of certain CA TLMS utilities by including the optional DATEFMT= parameter in the JCL for that utility, and specifying a different date format pattern as the DATEFMT value. If DATEFMT= is not included in the JCL, or is specified as DATEFMT=(DEFAULT), the value contained in member TLMSIPO is used.

This system option has the following format:

DATEFMT='*fmt*'

Specifies the date format for both input and output. The pattern specified must consist of all of the elements from any one group (1-3) listed next, and may also contain any combination of delimiters. Groups 1, 2 and 3 contain the elements of a complete date. They may be arranged in any order, but each element must be used once and only once.

Range: 10 characters

Delimiters: Slash (/), dash (-), comma (,) or blank. If a blank or comma is used, the parameter value must be enclosed in single quotes or parentheses.

Default: MM/DD/YYYY

Group 1 elements *YYYY DDD*

Group 2 elements *YYYY MM DD*

Group 3 elements *YYYY MMMDD*

YYYY

Year 1960 through 2155

DDD

Day 001 through 366

DD

Day 00 through 32

MM

Month 01 through 12

MMM

Month in character (JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC)

Separators	Examples	Results
period(.)	DDMMM.YYYY	01JUN.1995

Separators	Examples	Results
slash(/)	YYYY/DDD	1995/152
dash(-)	MM-DD-YYYY	06-01-1995
comma(,)	MMMDD,YYYY	JUN01,1995
space( )	DD MM YYYY	01 06 1995

Example: DATEFMT System Option

```
DATEFMT='MMMDD-YYYY'
```

DBLDRIV = Verify Same Device on Double Open for a Data Set

Use this system option to limit double opens of a data set to the same physical drive.

Note: Consider this option only if you allow double opens of a data set by the same job. Otherwise, it should be set to NO. Before setting this option, see the option DBLTIME= Allow/Disallow Double Opens for the Same Data Set (see page 48).

This system option has the following format:

```
DBLDRIV=[NO|YES]
```

NO

(Default) Verify that a double open of a data set is not on the same physical drive.

YES

Verify that a double open of a data set is on the same physical drive.

DBLTIME System Option—Allow/Disallow Double Opens for the Same Data Set

CA TLMS automatically prevents "double opens" of an output data set to the same volume by the same job without first being closed. To re-create a data set on the same volume on which it was originally created, set this option to a value greater than zero.

Use the DBLTIME system option to determine the time interval (after creation) during which CA TLMS allows a double open of the same tape data set by the same job.

If a data set requires a higher time than that set by this option, you must code SPACE=(1,(1,3)) on the DD statement for the data set. This will override the time limit set by the DBLTIME option.

Note: To limit the double open of a data set to the same physical device on which it was previously opened, use the DBLDRIV option.

This system option has the following format:

`DBLTIME=`*`hhmmss`*

hhmmss

Specifies the amount of time in hours, minutes, and seconds after creation that will pass, during which the same data set can be reopened.

The value 240000 permits all double opens. If specified as 000000, a SPACE=(1,(1,3)) will not override it.

Default: 000000 (disallow any double opens)

Example: DBLTIME System Option

This example illustrates that the same data set cannot be opened again after 12 hours from creation.

```
DBLTIME=120000
```

DEF System Option—Tape Default Retention by Data Set Name

Use the DEF system option to have a default keep date to be assigned based on the data set name. This option dynamically builds user table TLMSDTAB. No masking characters should be used in data set name definitions in the DEF option.

This system option has the following format:

DEF=[*dsname*|$DEFAULT$], [POS=*nn*], [DAYS=*nnn*]

DAYS=*nnn*

(Optional) Specifies the number of days the data set is to be retained.

Default: 7

DEF=*dsnname*

Specifies the data set name or prefix for retention. DSN=$DEFAULT$ generates a default KEEPDATE value for all data sets that do not match entries in the table.

POS=*nn*

(Optional) Specifies the starting position in a data set name.

Default: 1

A maximum of 128 DEF= statements may be supplied..

Examples: DEF System Option

```
DEF=DEF.AAA,DAYS=3
DEF=DEF.BBB
DEF=$DEFAULT$,DAYS=3 Base default assigned when no match is found)
```

This example uses the POS parameter with data set names.

```
PYR043.MONTHLY.DATA
PYR042.MONTHLY.DATA1
PYR055.MONTHLY.ROLL
```

The above data set names would be controlled by the following DEF definition:

```
DEF=MONTHLY,POS=8,DAYS=10
```

DISP System Option—Disposition Processing from JCL

Use the DISP system option to define the type of process for final disposition processing as specified in the JCL. This applies only to the controlling data set.

This system option has the following format:

`DISP=[YES|NO|OUTPUT]`

NO

Specifies that disposition is not processed and all volumes are KEEP.

OUTPUT

(Default) Specifies that disposition is for output only.

YES

Specifies that disposition processing from JCL is allowed.

EDM System Option—External Data Manager (EDM) Pointer

Use the EDM system option to define the two character suffix used to identify the active EDM member. It specifies the EDM rules used by CA TLMS to determine whether a volume is to be EDM controlled.

This system option has the following format:

`EDM=[xx|00]`

xx

Specifies a user-defined two-character alphanumeric or national (@,#,$) value.

00

(Default) Refers to CTOEDM00.

Example: EDM System Option

```
EDM=00
```

EDMEXIT System Option—Control the Exit (CTSUXEDM)

Use the EDMEXIT system option to determine whether the CA TLMS exit (TLMSXEDM) is active or inactive.

This system option has the following format:

EDMEXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CA TLMS exit is not activated.

YES

Specifies that the CA TLMS exit is activated. The exit name is CTSUXEDM.

exitname

Defines a name for the exit.

Example: EDMEXIT System Option

```
EDMEXIT=YES
```

FORSEC System Option—Security Checking for Foreign Volume Processing

Use the FORSEC system option to specifiy security checking on foreign volume processing when the SECURE option is set to YES.

This system option has the following format:

FORSEC=[NO|YES]

NO

(Default) Security checking is not activated for foreign volume processing.

YES

Security checking is activated for foreign volume processing.

FORSPEC System Option—Specific Request for Foreign Volume Processing

Use the FORSPEC system option to determine the restrictions for mounting a foreign volume for output.

This system option has the following format:

```
FORSPEC=[YES|NO]
```

YES

(Default) Requires you to code VOL=SER= for foreign volume processing.

NO

You can use a foreign volume for nonspecific requests.

IDSNVER System Option—Input Data Set Name Verification

Use the IDSNVER system option to specify whether the 44-character data set name is verified during OPEN for input processing.

This system option has the following format:

```
IDSNVER=[YES|NO]
```

YES

(Default) Verifies the input data set name. This is required when the SECOPN option is set to YES..

NO

Does not verify the input data set name.

Important! This option must be set to YES if the SECOPN option (security check for open tape processing) is set to YES.

INPUT System Option—Input Data Set Processing Requirements

Use the INPUT system option to determine following:

- If tapes used for input are reset to the data center
- If data sets are re-chained upon input
- If the VMF is updated for BLP data sets
- If the VMF is updated for nonlabeled data sets
- If the VMF is updated for standard labeled data sets

This system option has the following format:

```
INPUT=[YES|NO]
    [,CHN=YES|NO]
    [,BLP=YES|NO]
    [,NL=YES|NO]
    [,SL=YES|NO]
```

INPUT=YES|NO

Specifies if tapes used for input are reset to the data center.

Default: YES

CHN=YES|NO

(Optional) Specifies whether to chain data set upon input.

Default: NO

Note: Applications that stack multiple data sets and then read the data sets back in can cause overhead due to re-chaining upon input. We recommend that CHN= be set to NO

BLP=YES|NO

Specifies whether to update BLP data sets if the data set name in JCL does not match VMF.

Default: YES

NL=YES|NO

Specifies whether to update NL data sets if the data set name in JCL does not match VMF.

Default: YES

SL=YES|NO

Specifies whether to update SL data sets if the data set name in JCL does not match VMF.

Default: YES

Example: INPUT System Option

```
INPUT=YES,CHN=NO,NL=NO
```

INQACC System Option—Security Checking for Online Access

Determines whether security checking for online facilities is active or inactive. This is a global option which overrides individual options SECINQ and PROMPT.

This system option has the following format:

```
INQ=[NO|YES]
```

NO

Internal security checking is not activated for online access.

YES

Security checking is activated for online access.

KDATE System Option—Keep Date Determination

Use the KDATE system option to specify how a keep date is determined by indicating whether JCL, if coded, overrides the default retention. It also determines how calculated keep dates from the RMF will be handled.

Note: For information on keep date determination, see the *User Guide.*

This system option has the following format:

```
KDATE=[DEF|JCL|MAX]
```

DEF

The default retention will override EXPDT/RETPD in the JCL. TRS will use the calculated keep date if a count field is specified in the RMF entry.

See DEF= Tape Default Retention by Data Set Name for defining DTAB entries.

JCL

EXPDT/RETPD in JCL, if coded, will override the default retention. TRS will use the calculated keep date if a count field is specified in the RMF entry.

MAX

(Default) The larger of EXPDT in JCL or the default retention will be used as the keep date.

LBLDTRY System Option—Dynamic Label Changes During Realtime Processing

Use the LBLDTRY system option to determine whether dynamic label changes during realtime processing are allowed or disallowed.

This system option has the following format:

LBLDTRY=[NO|YES]

NO

(Default) Dynamic destruction of tape labels is not allowed.

YES

Dynamic destruction of tape labels is allowed.

LOGID System Option—Identification Number for SMF or Alternate Log Records

Use the LOGID system option to supply the identification number (record type code) assigned to the SMF or alternate log records (if RECOVRY=SMF or RECOVRY=ALTLOG; see RECOVRY= Create Backup Records).

This system option has the following format:

LOGID=*nnn*

nnn

Specifies the identification number assigned to the log records.

Range: 240 to 255

Default: 240

MANUAL System Option—Manual Update of Retention Schedules

Use the MANUAL system option to determine whether manual overrides of tape retention schedules are allowed. When manual overrides are allowed, the retention of the updated tapes may be changed from what was specified in the RMF.

This system option has the following format:

```
MANUAL=[NO|YES]
```

NO

(Default) The retention schedule for a tape may *not* be updated manually.

YES

The retention schedule for a tape may be updated manually.

NITEXIT System Option—Control the Exit (TLMSXNIT)

Use the NITEXIT system option to determine whether the CA TLMS tape initialization exit (TLMSXNIT) is active or inactive

This system option has the following format:

```
NITEXIT=[NO|YES|exitname]
```

NO

(Default) Specifies that the CA TLMS tape initialization exit is not activated.

YES

Specifies that the CA TLMS tape initialization exit is activated. The exit name is TLMSXNIT

exitname

Customer specified name for exit.

Example: NITEXIT System Option

```
NITEXIT=YES
```

NLSEC System Option—No Label (NL) Security Processing

Use the NLSEC system option to specify security checking for No Label (NL) processing when the SECURE option is set to YES.

This system option has the following format:

NLSEC=[NO|YES]

NO

(Default) Security checking is not activated for No Label processing.

YES

Security checking is activated for No Label processing.

NOTLMS System Option—CA TLMS NOT ACTIVE Processing

Use the NOTLMS system option to determine the job processing action to be taken at OPEN if CA TLMS is not active.

This system option has the following format:

NOTLMS=[ABEND|CONT]

ABEND

(Default) Unloads tape mounted if CA TLMS is not active.

CONT

If CA TLMS is not active, generate a message and continue normal processing.

NSLSEC System Option—Nonstandard Label (NSL) Security Processing

Use the NSLSEC system option to specify security checking for Nonstandard Label (NSL) processing when the SECURE option is set to YES.

This system option has the following format:

NSLSEC=[NO|YES]

NO

(Default) Security checking is not activated for Nonstandard Label processing.

YES

Security checking is activated for Nonstandard Label processing.

NSM System Option—Non Specific Mount Suffix

Use the NSM system option to specify the two character suffix used to identify the active CTONSM*xx* member.

This system option has the following format:

```
NSM=[00|xx]
```

00

Specifies the default and refers to CTONSM00.

xx

Is a 2 character user-defined suffix to CTONSM*xx*. user-defined two-character alphanumeric or national (@,#,$) value. This parameter specifies the scratch pool assignments for CA TLMS

Example: NSM System Option

```
NSM=00
```

OPNEXIT System Option—Control the OPEN Exit (TLMSXOPN)

Use the OPNEXIT system option to determine whether the CA TLMS OPEN exit (TLMSXOPN) is active or inactive.

Note: For information on this exit, see the section User Exits and Macros (see page 129).

This system option has the following format:

```
OPNEXIT=[NO|YES|exitname]
```

NO

(Default) Specifies that the CA TLMS OPEN exit is not activated.

YES

Specifies that the CA TLMS OPEN exit is activated. The exit name is TLMSXOPN

exitname

Customer specified name for exit.

Example: OPNEXIT System Option

```
OPNEXIT=YES
```

PAGESIZ System Option—Number of Lines per Printed Report Page

Use the PAGESIZ system option to specify the maximum number of lines on a printed report page.

This system option has the following format:

PAGESIZ=*nn*

nn

Identifies the number of lines per page for printed reports.

Default: 58

Example: PAGESIZ System Option

```
PAGESIZ=45
```

PROMPT System Option—User Logon ID Prompt

Use the PROMPT system option to determine whether the TSO interface will prompt the user for a CA TLMS logon ID. This option is dynamically set to NO when the INQACC option is set to YES.

This system option has the following format:

PROMPT=[NO|YES]

NO

A user will not be prompted for a TSO logon ID

YES

(Default) A user will be prompted for a TSO logon ID.

PROTECT System Option—Restrict Crash Protection to Selected Data Sets

Use the PROTECT system option to determine which output data sets will be protected by the crash protection feature of CA TLMS.

This system option has the following format:

PROTECT=[ALL|SELECT]

ALL

(Default) Crash protection will be applied to all data sets.

SELECT

Crash protection will be applied to only those data sets which have SPACE=(1,(1,2)) coded in their DD statements.

QSIZE System Option—Transaction Queue Entry Limit

Use the QSIZE system option to determine how many CA TLMS message queue entries will be formatted in common storage (CSA). Specify a high number if tape activity is high.

This system option has the following format:

QSIZE=*nnn*

nnn

Specifies the number of message queue entries to be formatted. If specified as 128, approximately 64K of CSA space is utilized.

Range: 0 to 128

Default: 64

Example: QSIZE System Option

```
QSIZE=60
```

RECOVRY System Option—Create Backup Records

Use the RECOVRY system option to determine whether to create backup records and where to write them. We recommend that you write them to the alternate log file to avoid the overhead required to process an entire SMF data set when recovering the CA TLMS Volume Master File.

This system option has the following format:

```
RECOVRY=[ALTLOG|SMF|NONE]
```

ALTLOG

(Default) Backup records will be written to the alternate log for recovery.

NONE

No backup records will be written.

SMF

Backup records will be written to the SMF SYS1.MANx files for recovery.

Example: RECOVRY System Option

```
RECOVRY=SMF
```

ROUTAUX System Option—Auxiliary Message Processing

Use the ROUTAUX system option to define the processing and display rules for auxiliary messages.

This system option has the following format:

```
ROUTAUX=[NO|YES]
    [,RT=nn]
    [,VSN=NO|YES]
    [,MSG=DEL|NDEL]
```

ROUTAUX=NO|YES

If NO is specified (and AUX= is unspecified), auxiliary messages are not supported and other parameters cannot be specified. YES indicates that auxiliary messages are supported.

If ROUTAUX=NO (or is not specified) and the AUX= option is specified, an options conflict will occur. If the ROUTAUX and AUX options are in conflict, an error message is issued and CA TLMS cannot properly initialize. If you no longer want to process auxiliary messages, the AUX= options should be removed from options member TLMSIPO.

RT=*nn*

(Optional) Specifies the number which identifies the route code for the auxiliary messages console.

Range: 01 to 16

Default: 02

VSN=NO|YES

(Optional) Determines if auxiliary messages contain the volume serial number.

Default: YES

MSG=DEL|NDEL

(Optional) Determines if auxiliary messages are non-deleteable.

Default: DEL

Example: ROTAUX System Option

```
ROUTAUX=YES,RT=10,VSN=YES,MSG=NDEL
```

ROUTINQ System Option—Inquiry Console Processing

Use the ROUTINQ system option to define the processing and display rules for the inquiry (librarian) console.

This system option has the following format:

```
ROUTINQ=[NO|YES]
    [,RT=nn]
    [,MSG=DEL|NDEL]
    [,UPD=YES|NO]
```

ROUTINQ=[NO|YES]

NO specifies that console inquiry is not supported. YES indicates that console inquiry is supported.

Default: YES

RT=*nn*

Is a number which identifies the route code for the inquiry console.

Range: 01 to 99

Default: 14

MSG=DEL|NDEL

NDEL specifies inquiry messages are non-deleteable.

Default: DEL

UPD=YES|NO

NO specifies the INQR task in inquiry mode. YES specifies INQR in update mode.

Default: YES

Example: ROUTNQ System Option

```
ROUTINQ=YES,RT=16,MSG=NDEL,UPD=YES
```

SCR System Option—Scratch Pool Suffix

Use the SCR system option to specify the two character suffix used to identify the active CTOSCRxx member.

This system option has the following format:

SCR=*xx*

xx

Defines a 2 character user-defined suffix to CTOSCRxx. This parameter specifies the scratch pool names assigned volume ranges for CA TLMS where *xx* is a user-defined two-character alphanumeric or national (@,#,$) value.

Default: 00 (Refers to CTOSCR00)

Example: SCR System Option

```
SCR=00
```

SCREXIT System Option—Control the Scratch Exit (TLMSRACF)

Use the SCREXIT control option to determine whether the CA TLMS scratch exit (TLMSRACF) is active or inactive.

This system option has the following format:

SCREXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CA TLMS scratch exit is not activated.

YES

Specifies that the CA TLMS scratch exit is activated. The exit name is TLMSRACF

exitname

Customer specified name for the exit

Example: SCREXIT System Option

```
SCREXIT=YES
```

SCRATCH System Option—Volume Scratch Control

Use the SCRATCH system option to determine whether volumes will be scratched by:

- TRS based on the controlling data set (CDS)
- The keep date expiration of all data sets along with all data sets that have been uncataloged from the OS/Catalog
- The keep date expiration of all data sets in the volume chain

This system option has the following format:

SCRATCH=[CDS|ALL]

ALL

Places all tapes to a type 3 control when the keep date for all data sets has expired and all data sets have been uncataloged.

CDS

(Default) Specifies that scratch volumes are based on expiration of the controlling data set.

Important! If the SCRATCH=ALL option is coded in TLMSIPO, tapes do not go scratch after their retention period has been completed unless all data sets on all volumes have met their KEEPDATES and have been uncataloged. SCRATCH=CDS causes the volume to be scratched and uncataloged when the retention period is complete. Types A, B, or C will protect multi-file volumes.

CA Technologies does not recommend SCRATCH=ALL. Assign retention types A, B, or C as needed.

Note: For more information on retention types, see the *User Guide*.

SECCLS System Option—Security Checking for DISP=YES or DISP=OUTPUT

Use the SECCLS system option determine whether the installed security system will be used to force security checking for DISP=YES or DISP=OUTPUT when the SECURE option is set to YES. (See the DISP option on DISP= Disposition Processing from JCL.)

This system option has the following format:

SECCLS=[NO|YES]

NO

(Default) Specifies that the installed security system will not be used to force security checking for DISP=YES or DISP=OUTPUT.

YES

Specifies that the installed security system will be used to force security checking on DISP=YES or DISP=OUTPUT.

SECEXIT System Option—Control the Security Exit (TLMSXSEC)

Use the SECEXIT system option to determine whether the CA TLMS security exit (TLMSXSEC) is active or inactive. If activating this exit, ensure that SECURE=YES. It is required that the exit TLMSXSEC is loaded before this option is set to YES.

This system option has the following format:

SECEXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CA TLMS security exit is not activated.

YES

Specifies that the CA TLMS security exit is activated. The exit name is TLMSXSEC

exitname

Customer specified name for the exit.

Example: SECEXIT System Option

```
SECEXIT=YES
```

SECINQ System Option—Security Checking for Online Inquiry

Use the SECINQ system option to determine whether the installed security system will be used to force security checking for online inquiries when the SECURE option is set to YES.

This system option has the following format:

```
SECINQ=[NO|YES]
```

NO

(Default) Specifies that the installed security system will not be used to force security checking for online inquiries.

YES

Specifies that the installed security system will be used to force security checking for online inquiries.

SECOPN System Option—Security Checking for Open Tape Processing

Use the SECOPN system option to determine whether the installed security system will be used to force security checking for open tape processing when the SECURE option is set to YES.

This system option has the following format:

```
SECOPN[NO|YES]
```

NO

Specifies that the installed security system will not perform security checking for open tape processing.

YES

Specifies that the installed security system will perform security checking for open tape processing

Important! If this option is set to YES, the IDSNVER option (input data set name verification) must be set to YES.

SECURE System Option—Global Security Option

Use the SECURE system option to determine whether the installed security system will be invoked. This is a global option that overrides any individual security option (INQACC, BLPSEC, NLSEC, FORSEC, NSLSEC, SECEXIT, SECOPN, SECCLS).

This system option has the following format:

SECURE=[NO|YES]

NO

(Default) Specifies that the installed security system will not be invoked. Individual security options will be ignored regardless of how they are specified.

YES

Specifies that the installed security system will be invoked, allowing the individual security system options to be in effect, as required.

When the option is displayed and YES is specified, the actual security system is displayed:

- S—RACF/SAF
- A—CA ACF2 Security
- T—CA Top Secret Security

SERIND System Option—Use of Out-of-Service Tapes

Use the SERIND system option to prevent reuse of an out-of-service/scratch-indicated tape by indicating whether CA TLMS will reset the out-of-service status to in-service when a tape is mounted for input. Out-of-service tapes cannot be used for output regardless of the option setting.

This system option has the following format:

SERIND=[NO|YES]

NO

Specifies that the service indicator is not reset to IN when the tape is opened for input.

YES

(Default) Specifies that type 4 service indicated tapes cannot be used for output. The service indicator is reset.

SMS System Option—SMS Interface

Use the SMS system option to determine whether the SMS interface is to be active. This parameter is used during pool assignment and verification. It is also used during OPEN for output processing to obtain SMS management class name for DSN being processed.

This system option has the following format:

SMS=[NO|YES]

NO

(Default) Specifies that the SMS interface is not to be active.

YES

Specifies that the SMS interface is to be active.

SPLEXIT System Option—Control the Command Exit (CTSUX1G)

Use the SPLEXIT system option to determine whether the CTS scratch subpool exit (CTSUX1G) is active or inactive.

This system option has the following format:

SPLEXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CTS scratch subpool exit is not activated.

YES

Specifies that the CTS scratch subpool exit is activated. The exit name is CTSUX1G

exitname

Customer specified name for the exit.

Example: SPLEXIT System Option

```
SPLEXIT=YES
```

TRSEXIT System Option—Control the Tape Retention Exit (TLMSXTRS)

Use the TRSEXIT system option to determine whether the CA TLMS tape retention exit (TLMSXTRS) is active or inactive.

This system option has the following format:

TRSEXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CA TLMS retention exit is not activated.

YES

Specifies that the CA TLMS retention exit is activated. The exit name is TLMSXTRS

exitname

Customer specified name for the exit.

Example: TRSEXIT System Option

TRSEXIT=YES

More Information:

TLMSXTRS - Tape Retention System Exit (see page 141)

UNCATLG System Option—Uncatalog Action When Scratch

Use the UNCATLG system option to specify how the data sets on a volume will be uncataloged when the volume reaches SCRATCH status.

This system option has the following format:

UNCATLG=[NO|YES]

NO

Specifies do not uncatalog the data sets when a volume reaches SCRATCH status.

YES

(Default) Specifies to automatically uncatalog the data sets when a volume reaches SCRATCH status.

More Information:

TLMSXUPD - Volume Master File Update Exit (see page 142)

UPDEXIT System Option—Control the Volume Master File Update Exit (TLMSXUPD)

Use the UPDEXIT system option to determine whether the CA TLMS Volume Master File update exit (TLMSXUPD) is active or inactive.

This system option has the following format:

UPDEXIT=[NO|YES|*exitname*]

NO

(Default) Specifies that the CA TLMS file update exit is not activated.

YES

Specifies that the CA TLMS file update exit is activated. The exit name is TLMSXUPD.

exitname

Customer specified name for the exit.

Example: UPDEXIT System Option

```
UPDEXIT=YES
```

USER System Option—Internal User Security

Use the USER system option to create the TLMSUTAB.

The TLMSUTAB table contains a list of userids and their access level. The userids are used by TLTPISPF when external security is not active (SECURE=NO). When external security is active, TLMSUTAB allows the user to bypass the DSN security calls. Users like librarians can access all datasets in the VMF without update access in the external security system for all datasets.

Important! Any user that is not defined in this table is automatically assigned the access that is specified in the last entry as a default. The access that is assigned in the last entry should always be ACCESS=I or ACCESS=N if security is active (INQACC=YES).

This option has the following format:

USER=*userid* PWD=*password* ACCESS=[N|I|U]

USER=*userid*

Specifies the userid.

Valid values: 1 to 8 alphanumeric characters

PWD=*password*

Specifies the password for this userid.

Valid values: 1 to 8 alphanumeric characters or blank. (Password must contain one alpha character.)

Default: blank

ACCESS=[N|I|U]

Specifies the access level for the userid.

- U=update. User can update VMF.
- I=inquiry. User can browse VMF.
- (Default) N=none. No authority given.

Supply a maximum of 100 USER= statements.

Examples: User System Option

```
USER=JACK,PWD=MYPWD,ACCESS=U

USER=JILL,ACCESS=I

USER=DEFAULT,ACCESS=N
```

VMFINAM System Option—VMF Index Data Set Name

Use the VMFINAM system option to specify the name of the VMF Index. This name is used to verify that batch programs are processing the active VMF Index. This value is overridden by the data set name in the CTS JCL when CA TLMS is started. This entry should be omitted unless needed. This parameter is only needed to run CA TLMS programs after TLMSRIM has run but before CA TLMS is started under CTS.

This system option has the following format:

`VMFINAM=`*`dsn`*

dsn

Specifies the fully qualified data set name of the VMF Index.

Example: VMFINAM System Option

```
VMFINAM=CAI.TLMS.VMFINDEX
```

VMFNAM System Option—VMF Data Set Name

Use the VMFNAM system option to specify the name of the VMF. This name is used to verify that batch programs are processing the active VMF. This value is overridden by the data set name in the CTS JCL when CA TLMS is started. This entry should be omitted unless needed. This parameter is only needed to run CA TLMS programs after TLMSRIM has run but before CA TLMS is started under CTS.

This system option has the following format:

`VMFNAM=`*`dsn`*

dsn

Specifies the fully qualified data set name of the VMF

Example: VMFNAM System Option

```
VMFNAM=CAI.TLMS.VMF
```

VSNPAD System Option—Volume Serial Numbers of Less Than Six Digits

Use the VSNPAD system option during VMF initialization to permit CA TLMS to process volume serial numbers of less than six digits.

This system option has the following format:

VSNPAD=*x*

x

Specifies the alphanumeric pad character (special characters are disallowed). The volume serial number is right justified and padded on the left with the specified character.

Example: VSNPAD System Option

```
VSNPAD=A
```

VSNREQD System Option—Volume Serial Number Required at OPEN

Use the VSNREQD system option to determine whether you want to require that the operator supply a volume serial before a volume is opened for output. This option applies when CA TLMS is unable to determine the volser of the output tape. Normally, this indicates a NL tape was mounted.

This system option has the following format:

VSNREQD=[NO|YES]

NO

Specifies that If NO is specified, CA TLMS still attempts to find a VSN in the JCL and verify that the volume is scratch. However, if it cannot be found, it will allow the mounted volume to be used without prompting the operator for the VSN.

YES

(Default) Specifies that If the VSN is supplied in the JCL or the data set is cataloged, CA TLMS will verify that the volume is scratch. If no VSN can be found, CA TLMS prompts the operator to input the VSN for scratch verification.

Example: VSNREQD System Option

```
VSNREQD=NO
```

Chapter 3: Common Tape System

This section contains the following topics:

About the Common Tape System

The Common Tape System (CTS) is a component distributed with CA TLMS for the purpose of creating a single image and common interfaces for CA z/OS tape management products (CA TLMS and CA 1). This allows other CA products, client programs, and other vendor products to have a single set of interfaces to CA tape solutions. It consists of utility programs, macros, and a subtasking address space.

This section deals with the subtasking address space. The CTS address space consists of a main task, a command task, a dump task, and service programs. An optional external (gummed) label task is part of CTS but is described in another section. TLMS is not part of CTS but has been made CTS compliant. CTS replaces the functions provided by OMS in prior versions.

CTSMAIN is the main task of the CTS address space. It maintains the other task as subtasks and anchors the service programs. The service programs provide for inter-task communications, logging messages, and snap dumps. Each subtask has a 1 to 4 character TASKID that is unique within the address space. When CTSMAIN attaches a subtask, it creates a control block (CTSB) which contains the TASKID, control data, and parameters to pass to the subtask. The address of the CTSB is passed to the subtask. A CTS compliant task, communicates with other CTS tasks on a peer-to-peer basis through their CTSBs. If a task abends, CTSMAIN will restart it as many times as was defined in its CTSB.

There are several tasks which are predefined to CTSMAIN. All other tasks must be defined through a SET TASK(....) command before they can be started with a START command. CMD, DUMP, API, DBS, and LAB are the predefined task IDs. CMD and DUMP are discussed below, and LAB in a later section. API and DBS are reserved for future services.

CTSCMD is the command processing task for the CTS address space. It is the only task that communicates with CTSMAIN. Its TASKID is CMD. CMD is predefined to CTSMAIN and is automatically started by CTSMAIN. The function of CMD is to receive commands from multiple sources, verify, and execute them. CTSCMD gets commands from five sources. First a command can be entered in the PARM= field of the EXEC JCL statement; it is processed first. Next CTSCMD reads the command from the CTSSYSIN DD JCL statement. Usually this points to the CAI.CTAPOPTN member CTSSTART. After that, commands are processed from any of the remaining sources. Commands can be sent to CMD from any other CTS task. Commands may come from a z/OS modify for CTS (f CTS, DISPLAY ACTIVE). A command can be entered in reply to the optional outstanding WTOR. CTSCMD treats commands from all sources equally.

CTSDMP is the other predefined and automatically started CTS task. Its TASKID is DUMP. Its function is to automatically copy dumps to a SYSOUT so that they are available without having to stop the CTS address space. After CTSDMP is started, it checks for the presence of a SYSUDUMP or SYSABEND allocated to DASD. If the DD is missing or is for a SYSOUT, the DUMP task terminates with an RC=0 since it is not needed to unspool dumps. If there is a dump data set, CTSDMP attempts to copy a dump from it and notify the console when it is complete. Because the dump data set is defined as MOD, more dumps may be added while it is being copied by dump. CTSDMP will empty the data set when it completes.

Important! Ensure that the dump data set has the proper attributes for a dump data set and that it has been written to before CTSDMP is started, if not an abend will occur.

CTSMSG is a service program that allows messages from multiple tasks in the CTS address space to log messages to a single log. It will copy that log to SYSOUT when the SPINOFF LOG command is issued. No response is given.

CTSSNP is a service program that allows snap dump from multiple tasks in the CTS address space to be sent to a single file. It will copy that file to SYSOUT when the SPINOFF SNAP command is issued. No response is given.

All other CTS tasks must be defined to CTS by SET commands and then attached through the START command. The CTS address space only provides a generic multi-tasking environment and the basic messages and dump service.

The purpose of the Common Tape System (CTS) is to provide a set of interfaces to all CA Tape Management products. CTS consists of a started task, utilities, and interface programs. CTS is started from SYS1.PROCLIB(CTS). During startup it reads commands from the CTSSYSIN DD, which is CAI.CTAPOPTN (CTSSTART) by default. This section contains a discussion of those commands.

CTSDMP is the other predefined and automatically started CTS task. Its TASKID is DUMP. Its function is to automatically copy dumps to a SYSOUT so that they are available without having to stop the CTS address space. After CTSDMP is started, it checks for the presence of a SYSUDUMP or SYSABEND allocated to DASD. If the DD is missing or is for a SYSOUT, the DUMP task terminates with an RC=0 since it is not needed to unspool dumps. If there is a dump data set, CTSDMP attempts to copy a dump from it and notify the console when it is complete. Because the dump data set is defined as MOD, more dumps may be added while it is being copied by dump. CTSDMP will empty the data set when it completes.

Important! Ensure that the dump data set has the proper attributes for a dump data set and that it has been written to before CTSDMP is started, if not an abend will occur.

CTSMSG is a service program that allows messages from multiple tasks in the CTS address space to log messages to a single log. It will copy that log to SYSOUT when the SPINOFF LOG command is issued. No response is given.

CTSSNP is a service program that allows snap dump from multiple tasks in the CTS address space to be sent to a single file. It will copy that file to SYSOUT when the SPINOFF SNAP command is issued. No response is given.

All other CTS tasks must be defined to CTS by SET commands and then attached through the START command. The CTS address space only provides a generic multi-tasking environment and the basic messages and dump service.

The purpose of the Common Tape System (CTS) is to provide a set of interfaces to all CA Tape Management products. CTS consists of a started task, utilities, and interface programs. CTS is started from SYS1.PROCLIB(CTS). During startup it reads commands from the CTSSYSIN DD, which is CAI.CTAPOPTN (CTSSTART) by default. This section contains a discussion of those commands.

Initialization

The following example shows the commands used to start the CTS task:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* CA-COMMON TAPE SYSTEM STARTUP COMMANDS                             */
/* CA TLMS STARTUP PARAMETERS                                         */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* SET OPTIONS                                                        */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
SET WTOR /* REQUEST HOT (OUTSTANDING) WTOR                           */
SET NOSNAP /* REQUEST DYNOSTIC SNAP DUMPS                           */
SET LOG /* REQUEST COMMAND AND MESSAGE LOGGING                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* DEFINE TASK NORMALLY USED                                          */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
SET TASK(TLMS) PGM(TLMSMAIN) RETRY(*) PARM() /* TLMS                 */
SET TASK(INQR) PGM(TLMSINQR) RETRY(*) PARM() /* INQR/UPDATE         */
SET TASK(DTS) PGM(CTSDTS)
SET TASK(SCHD) PGM(CTSSCHD)
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* AUTOMATICALLY START THESE TASK                                     */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
START TLMS /* TAPE MANAGEMENT                                        */
/*START INQR /* INQUIRY / UPDATE TASK                               */
/*START LAB /* EXTERNAL LABEL PRINT TASK                           */
/*START DTS /* DISTRIBUTED TAPE SUPPORT                            */
/*START SCHD /* CTS SCHEDULER                                       */
```

SET LOG

Indicates that CTS retains a message log.

SET NOSNAP

Indicates that the option for recording snap dumps is turned off.

SET NOWTOR

Indicates that the optional outstanding CA$F000R WTOR is not issued. Commands are entered through the console MODIFY command instead.

START LAB

Indicates that the Tape management Online Label Interface (LAB) is to be attached.

START TLMS

Indicates that the Tape Management system is to be attached.

Commands

The CTS address space is created by starting the CTS task. The CTS task is a subtasking supervisor, which provides support for CTS Services. It provides the means for starting, stopping, and controlling CTS subtasks. The CTS task also provides display and diagnostic facilities.

- The CTS task processes commands from console modify, CAI.CTAPOPTN member CTSSTART, EXEC PARM=, CTS internal message facility, and reply to an outstanding WTOR.
- Commands from any source are free-form in MACRO style and/or TSO style syntax.
- The syntax is a command-word followed by parameters separated by either blanks or commas.
- TSO style comments (/*.....*/) may appear anywhere in the command line or alone on the command line.
- Parameters can be keyword, positional or value, and they can be specified in any order.
- Keyword parameters can be of either the KWD= or KWD( ) style.
- Values are constants like SETUP, and can be specified as the alternate form NOSETUP.
- Numbers specified for positional and keyword parameters can be from 1- to 15-digits with any number of leading zeros.
- Parameters which contain blanks, commas or special characters must be enclosed in '.....' or (.....).

Command Descriptions

CANCEL

Request a CTS subtask to ABEND with a user 111 and dump.

DISPLAY or D

Request a display of CTS options, defined tasks, or active tasks.

FORCE

Detach a CTS subtask causing its forceable termination.

MSG or M

Send text to a CTS subtask.

SET

Set one or more CTS options, or a CTS subtask definition.

SPINOFF

Release SYSOUT for printing.

START or S

Start a CTS subtask for the task-ID. If the task ID is defined to CTS the task ID is the only parameter required. If not defined to CTS the PGM parameter must be used to identify the program to attach. The other parameters may be used to specify or override defined values.

STATUS

Request subtask to display status information.

STOP or P

Request a subtask to complete normally.

The parameter and description for each command is as follows:

CANCEL Command

Parameter	Description
tid or ALL	Task ID. One to four characters must be unique to CTS address space. If ALL is specified a CANCEL command is sent to every CTS subtask. Only those tasks which have had a START command issued may be canceled.

DISPLAY Command

Parameter	Description
OPTS or O	Display CTS options.
TASKS or T	Display all tasks defined to CTS.
ACTIVE or A	Display all currently active subtasks.

FORCE Command

Parameter	Description
tid	Task ID. One to four characters must be unique to CTS address space. Required. Only those tasks which had a START command may be forced.

MSG Command

Parameter	Description
tid or ALL	Task ID. One to four characters must be unique to CTS address space. If ALL is specified the text command is sent to every CTS subtask.
text	Text to be sent to subtask. If this text contain blanks, commas or special characters it must be enclosed in '.....' or (.....).

SET Command for Options

Parameter	Description
WTOR or NOWTOR	Option for outstanding WTOR. If option is WTOR, then CTS keeps a WTOR issued to receive operator commands.
SNAP or NOSNAP	Option for recording diagnostic snap dumps. If the option is SNAP, the dumps are recorded. This can be printed by using the SPINOFF SNAP command.
LOG or NOLOG	Option for recording message log. If the option is LOG, a log of messages are retained. This can be printed by using the SPINOFF LOG command.

SET Command to Define Subtasks

Parameter	Description
tid	Task ID. Required. One to four characters must be unique to CTS address space.
PGM(.....)	Program name. Required. This is the name of the program to be attached.

Parameter	Description
PARM(...)	Data to be sent to the subtask via internal message. Default is no data.
RETRY(...)	The number of times a subtask is to be restarted after completing with a nonzero completion code. Default is 0.

START Command

Parameter	Description
Task ID	Required. One to four characters must be unique to CTS address space.
PGM(nnnnnnnn)	Program name. Required if task not defined to CTS. This is the name of the program to be attached.
PARM(nnn)	Data to be sent to the subtask by internal message. Default is no data.
RETRY(nnn)	The number of times a subtask is to be restarted after completing with a nonzero completion code. Default is 0.

SPINOFF Command

Parameter	Description
tid	Task ID. One to four characters must be unique to CTS address space. Sends a SPINOFF command to a subtask.

Example

```
F CTS,SPINOFF,SNAP
              LOG
              WTOR
```

STOP Command

Parameter	Description
tid or ALL	Task ID. One to four characters must be unique to CTS address space. If ALL is specified a STOP command is sent to every CTS subtask. Only those tasks which have had a START command issued may be stopped.

STATUS Command

Parameter	Description
tid or ALL	Task ID. One to four characters must be unique to CTS address space. If ALL is specified a STATUS command is sent to every CTS subtask. Each task which can report status, displays and/or logs its status. The format and content of status varies by task. Status from the CMD task is the same as issuing a “Display OPT” and “Display Active”.

Command Samples

CANCEL TMAP

The TMAP subtask abends with a user 111 and a dump. If the DUMP subtask is active, the abend dump will be spun off to SYSOUT.

START TLMS RETRY(4)

The TLMS subtask is attached and is restarted the next four times it ends with a nonzero return code. If TLMS ends with a zero return code, it will not be restarted.

MSG TLMS TRACE

The text TRACE is sent to the TLMS subtask. TRACE tells TLMS to snap all transactions to the SNAP log on entry and exit to TLMS.

MSG INQR 'UPV VOL001 SCRATCH=YES'

The text is sent to the INQR subtask which will process it just as though it had been entered in response to INQRs WTOR. In this case, VOL001 is scratched.

SET LABDEBUG(1)

The DEBUG switch in LABs CTSB is set to 1. The meaning of the DEBUG setting varies with the task. Use only when instructed by CA TLMS Support.

SET LAB DEBUG()

The DEBUG switch in LABs CTSB is set to reset to blank.

STOP ALL

STOP will be sent to all subtasks in the CTS address space. Each subtask terminates with a zero return code. The one exception is CMD. CMD never terminates while another subtask is active, because it is needed to process the STOP or CANCEL for the other task. If CMD receives a command to terminate when another task is active, waits 20 seconds then tries again. If another task is still active, it issues a message and remains active.

The following example displays how to turn on a CTS trace and where the output is routed:

```
A) F CTS,SET SNAP
B) F CTS,MSG TLMS,TRACE   (run job)
C) F CTS,SPINOFF,SNAP     (after job completes)
D) F CTS,MSG TLMS,NOTRACE
```

CTS Started Task

The following member is CTS in CAI.CTAPPROC.

```
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*            CA COMMON TAPE SYSTEM                                    *
//*                 FOR TLMS                                            *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*
//*
//CTS      PROC PRM=,                       /*  EXEC TIME COMMAND *
/
//             ALOG='CAI.TLMS.ALOG'
//*            EARLLIB='CAI.CTAPEARL',     /* OPTIONAL */
//*            EARLMAC='CAI.CTAPECPB',     /* OPTIONAL */
//             CTSDMP='CAI.CTS.DUMP',
//             LAB1='A',
//             LAB2='A',
//             LAB3='A',
//*            LBL='TLMSLBLS'
//             LINKLIB='CAI.CTAPLINK',
//*            SPIN='A',
//             START='CAI.CTAPOPTN(CTSSTART)',
//             VMF='CAI.TLMS.VMF',
//*            VMFIDX='CAI.TLMS.VMFINDEX',
//*            WORK='SYSDA',
//             SOUT='A'
//*
//CTSGO    EXEC PGM=CTSMAIN,REGION=3M,IME=1439,
//         PARM='&PRM'.
//*
//SPINOFF  OUTPUT CLASS=&SPN.              SYSOUT FOR SPINOFF
//OUTPUTP1 OUTPUT CLASS=&LAB1.             LABEL PRINTER #1
//OUTPUTP2 OUTPUT CLASS=&LAB2.             LABEL PRINTER #2
//OUTPUTP3 OUTPUT CLASS=&LAB3.             LABEL PRINTER #3
//*
//STEPLIB  DD DISP=SHR,DSN=&LINKLIB.
//SYSOUT   DD SYSOUT=&SOUT.
//SYSPRINT DD SYSOUT=&SOUT.
//SYSUDUMP DD DISP=MOD,DSN=&CTSDMP.        DUMP SPOOL
//*ABNLIGNR DD DUMMY                       DISABLE ABENDAID
//CTSSYSIN DD DISP=SHR,DSN=&START.         STARTUP COMMANDS
//*
```

```
//*----------- TLMS                                    (REQUIRED)
//CAIVMF    DD DISP=SHR,DSN=&VMF.          TAPEDB (UPDATE)
//CAIVMFI   DD DISP=SHR,DSN=&VMF.          TAPEDB (INPUT)
//TAPEDB    DD DISP=SHR,DSN=&VMF.          TAPEDB (CTSDBIO)
//*
//*----------- TLMS                                    (OPTIONAL)
//*CAIALOG  DD DISP=SHR,DSN=ALOG           ALTERNATE TRAN LOG
//*CAIVMFXI DD DISP=SHR,DSN=&VMFIDX.       TAPEDB INDEX (INPUT)
//*CAIVMFXU DD DISP=SHR,DSN=&VMFIDX.       TAPEDB INDEX (UPDATE)
//*


//*----------- EXTERNAL TAPE LABELS                (OPTIONAL)
//*EARLOBJ  DD UNIT=&WORK.,                  EARL COMPILE OUTPUT
//*            SPACE=(TRK,2),DISP=(NEW,DELETE)
//*EARLLIB DD DISP=SHR,DSN=&EARLLIB.         EARL SOURCE LIB
//*SORTIN   DD DUMMY
//*SORTOUT  DD DUMMY
//*SYSIN    DD DISP=SHR,DSN=&EARLLIB(.&LBL). LABEL FRMTTR SOURCE
//*
//*
```

Online Label Interface

External gummed labels are generated through the Online Label Interface (LAB) subtask of CTS.

Gummed labels produced by CA TLMS use the Online Label Interface (LAB) to generate labels.

The Online Label Interface is made up of three components:

- The input processor (CTSLBLIN) which is used to obtain each input record and pass the information to the appropriate EARL program.
- EARL which is used to process the input records, format the label image and call the output processor. Member TLMSLBLS in CAI.CTAPEARL is the EARL example provided to generate the default gummed labels for CA TLMS. This member can be copied and modified to customize labels for your site.
- The output processor (CTSLBLOT) which is used to write a formatted label image to one or more output locations.

Processing Steps

The CTS procedure provided with this version uses member TLMSLBLS in CAI.CTAPEARL to create labels. The following example shows the code contained in the TLMSLBLS member with a discussion of the various sections. If modification of this is required to customize labels for your site, you can modify this code or make a copy for modification. Verify that the LBL parameter in the CTS procedure points to the name of the member that you wish to use.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  SAMPLE OUTPUT LABELS FOR REELS AND CARTRIDGES                              !
!                                                                             !
!  THE FOLLOWING FORMATS ARE BEING PRODUCED IN THIS SAMPLE:                   !
!                                                                             !
!  REELS:                                                                     !
!     ....+....1....+....2....+....3....+....4....+                           !
!                                                                             !
!  1  DSN.....                                                                !
!  2  JOBNAME. STEPNAME HH&COLON.MM&COLON.SS YYDDD UUUU DEN.                  !
!  3  CPU. LRECL BLKSI. RFM. BLKCNT.... ERG FSN...                            !
!  4  VOLSER # VLSQ/VCNT T X       JOBACT.....                                !
!                                                                             !
!  CARTS:                                                                     !
!     ....+....1....+....2....+....                                           !
!                                                                             !
!  1  DSN.....                                                                !
!  2   DSN2.............. JOBNAME.                                            !
!  3   VOLSER STEPNAME HH&COLON.MM&COLON.SS                                   !
!  4   YYDDD UUUU DEN. X                                                      !
!  5   LRECL BLKSI. RFM. VSEQ/VCNT                                            !
!  6   BLKCNT.... FSN... # T ERG.                                             !
!  7   JOBACCT.....   CPU.                                                    !
!                                                                             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 COPY LABELDEF    2
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  OUTPUT LABEL IMAGE  10 ROWS BY 80 COLUMNS                !   3
!                                                           !
!  "ROWS" DEFINE THE NUMBER OF LINES IN EACH LABEL FROM     !
!  THE FIRST LINE OF ONE LABEL TO THE FIRST LINE IN THE     !
!  NEXT LABEL.                                              !
!                                                           !
!  "COLUMNS" DEFINE THE NUMBER OF CHARACTERS IN EACH ROW.   !
!                                                           !
!  "LINE" DEFINES THE INDIVIDUAL ROW IN THE ARRAY.          !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 DEF OUTPUT_LABEL (X 802) = ' ' 'OUTPUT' 'LABEL'   4

 DEF ROWS           = OUTPUT_LABEL  001-001 B 'ROWS'
 DEF COLUMNS        = OUTPUT_LABEL  002-002 B 'COLUMNS'
 DEF LINE  ARRAY 10 = OUTPUT_LABEL  003-082 X      5

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!
! THE FOLLOWING ARE ALTERED PRINT FIELDS.
    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!

 DEF ACCT_P           = ACCT            001-012 X  6
 DEF LABEL_P          = LABEL_TYPE      001-001 X
 DEF DSN_P1           = DSN             001-025 X
 DEF DSN_P2           = DSN             026-044 X
 DEF F_CTIME (N 6.0)  = NONE
 DEF F_CTIME_HH       = F_CTIME         001-002 N
 DEF F_CTIME_MM       = F_CTIME         003-004 N
 DEF F_CTIME_SS       = F_CTIME         005-006 N
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  WORK LINES FOR LABEL REEL IMAGE - 80 COLUMNS WIDE !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 DEF WORK_LINE (X 80) = ' '

 DEF REEL01_DSN     = WORK_LINE     001-044 X      7

 DEF REEL02_CJOB    = WORK_LINE     001-008 X
 DEF REEL02_CSTEP   = WORK_LINE     010-017 X
 DEF REEL02_CTIMEHH = WORK_LINE     019-020 N
 DEF REEL02_CTIMED1 = WORK_LINE     021-021 X
 DEF REEL02_CTIMEMM = WORK_LINE     022-023 N
 DEF REEL02_CTIMED2 = WORK_LINE     024-024 X
 DEF REEL02_CTIMESS = WORK_LINE     025-026 N
 DEF REEL02_CDATE   = WORK_LINE     029-033 N
 DEF REEL02_CUNIT   = WORK_LINE     035-038 X
 DEF REEL02_DEN     = WORK_LINE     040-043 X

 DEF REEL03_CPUID   = WORK_LINE     001-004 X
 DEF REEL03_LRECL   = WORK_LINE     006-010 N
 DEF REEL03_BLKSIZE = WORK_LINE     012-017 N
 DEF REEL03_RECFM   = WORK_LINE     019-022 X
 DEF REEL03_BLKCNT  = WORK_LINE     024-033 N
 DEF REEL03_ERG     = WORK_LINE     035-037 N
 DEF REEL03_FILESEQ = WORK_LINE     039-043 N

 DEF REEL04_VOLSER  = WORK_LINE     001-006 X
 DEF REEL04_FORGN   = WORK_LINE     008-008 X
 DEF REEL04_VOLSEQ  = WORK_LINE     010-013 N
 DEF REEL04_VOLCNT  = WORK_LINE     015-017 N
 DEF REEL04_LABEL   = WORK_LINE     019-019 X
 DEF REEL04_TRTCH   = WORK_LINE     021-024 X
 DEF REEL04_ACCT    = WORK_LINE     029-040 X
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  WORK LINES FOR LABEL CART IMAGE - 80 COLUMNS WIDE !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


 DEF CART01_DSN     = WORK_LINE     001-025 X

 DEF CART02_DSN2    = WORK_LINE     002-020 X
 DEF CART02_CJOB    = WORK_LINE     022-029 X

 DEF CART03_VOLSER  = WORK_LINE     002-007 X
 DEF CART03_CSTEP   = WORK_LINE     009-016 X
 DEF CART03_CTIMEHH = WORK_LINE     019-020 N
 DEF CART03_CTIMED1 = WORK_LINE     021-021 X
 DEF CART03_CTIMEMM = WORK_LINE     022-023 N
 DEF CART03_CTIMED2 = WORK_LINE     024-024 X
 DEF CART03_CTIMESS = WORK_LINE     025-026 N

 DEF CART04_CDATE   = WORK_LINE     002-006 N
 DEF CART04_CUNIT   = WORK_LINE     008-011 X
 DEF CART04_DEN     = WORK_LINE     013-016 X
 DEF CART04_TRTCH   = WORK_LINE     018-021 X

 DEF CART05_LRECL   = WORK_LINE     002-006 N
 DEF CART05_BLKSIZE = WORK_LINE     008-013 N
 DEF CART05_RECFM   = WORK_LINE     015-018 X
 DEF CART05_VOLSEQ  = WORK_LINE     020-023 N
 DEF CART05_VOLCNT  = WORK_LINE     025-028 N

 DEF CART06_BLKCNT  = WORK_LINE     002-011 N
 DEF CART06_FILESEQ = WORK_LINE     013-017 N
 DEF CART06_FORGN   = WORK_LINE     020-020 X
 DEF CART06_LABEL   = WORK_LINE     022-022 X
 DEF CART06_ERG     = WORK_LINE     024-027 N

 DEF CART07_ACCT    = WORK_LINE     002-013 X
 DEF CART07_CPUID   = WORK_LINE     018-021 X
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! THE FOLLOWING ROUTINE WILL GENERATE AND PRINT 10 CART SETUP LABELS  !
! USING THE DEFAULT OUTPUT IF THE VALUE IN 'SETUP_REQUESTED' = 'Y'.   !8
! IF REEL SETUP LABELS ARE PREFERRED, CHANGE THE VALUE IN THE         !
! 'SETUP_TYPE' FIELD TO 'REEL'.                                       !
! IF MORE THAN 10 SETUP LABELS ARE DESIRED, CHANGE THE VALUE IN THE   !
! 'SETUP_MAXIMUM' FIELD TO THE DESIRED VALUE.                         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 DEF SETUP_REQUESTED      (X 1)   = 'Y'
!DEF SETUP_TYPE           (X 4)   = 'CART'
!DEF SETUP_TYPE           (X 4)   = 'REEL'
 DEF SETUP_TYPE           (X 4)   = 'BOTH'
 DEF SETUP_COUNT          (P 2.0) = 0
 DEF SETUP_MAXIMUM        (P 2.0) = 10
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! PRINT SETUP LABELS                                                   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 IF SETUP_REQUESTED  = 'Y'
     SET SETUP_REQUESTED = 'N'                        9
     IF SETUP_TYPE = 'REEL' OR SETUP_TYPE = 'BOTH'
        SET SETUP_COUNT = 0
        PERFORM SETUP_REEL
     ENDIF
     IF SETUP_TYPE = 'CART' OR SETUP_TYPE = 'BOTH'
        SET SETUP_COUNT = 0
        PERFORM SETUP_CART
     ENDIF
 ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! THE FOLLOWING STATEMENTS READ THE NEXT INPUT RECORD                   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
GETTAG:                                              10
 GET TAPEDB
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! IF END-OF-FILE IS REACHED - CALL CTSLBLOT TO CLOSE ALL FILES          !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 IF TAPEDB = 'E'
    SET REQUEST = 'C'                                11
    CALL CTSLBLOT USING LABEL_RECORD COMM_AREA
    GOTO EOJ
 ENDIF
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! IF A SPINOFF REQUEST WAS ENTERED - CALL CTSLBLOT WITH THE REQUEST   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 IF REQUEST = 'S'
    CALL CTSLBLOT USING LABEL_RECORD COMM_AREA    12
    GOTO GETTAG
 ENDIF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!>>>>>>   B Y P A S S   O R    C H A N G E   L A B E L         <<<<<<!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  REQUEST = 'L'    CONTAINS PREFORMATTED DATA
!  REQUEST = 'V'    CONTAINS DATABASE RECORD
!  REQUEST = 'D'    CONTAINS DATABASE RECORD    13
!                   IGNORE ALL OTHER TYPES
!====>  USER CODE GOES HERE

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! IF A LABEL IMAGE IS READ - CALL CTSLBLOT TO PRINT THE LABEL IMAGE   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 IF REQUEST = 'L'
    PERFORM SELECT_IMAGE_OUTPUT
    CALL CTSLBLOT USING LABEL_RECORD COMM_AREA
    GOTO GETTAG
 ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! IF A VOLUME OR DATA RECORD IS READ AND IT CONTAINS A CARTRIDGE      !
! DENSITY, FORMAT A LABEL IMAGE FOR CARTRIDGES.                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 IF REQUEST = 'V' OR REQUEST = 'D'
    IF DEN = '38K' OR DEN = '38KC' OR DEN = 'CART' OR DEN = 'IDRC'
       OR '3590'
       PERFORM SELECT_CART_OUTPUT
       PERFORM CART_LABEL
    ELSE
       PERFORM SELECT_REEL_OUTPUT
       PERFORM REEL_LABEL
    ENDIF
    GOTO GETTAG
 ENDIF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!>>>>>>>>>>>>>>>>>>> END OF MAIN LINE CODE <<<<<<<<<<<<<<<<<<<<<<<<<<<!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 GOTO GETTAG
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!      FORMAT AND OUTPUT A CART TYPE LABEL                               !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 CART_LABEL: PROC

    SET OUTPUT_LABEL   = ' '                    14
    SET ROWS           = 10
    SET COLUMNS        = 80
    SET WORK_LINE      = ' '
    SET CART01_DSN     = DSN_P1
    SET LINE(/1/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART02_DSN2    = DSN_P2
    SET CART02_CJOB    = CJOB
    SET LINE(/2/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART03_VOLSER  = VOLSER
    SET CART03_CSTEP   = CSTEP
    SET F_CTIME        = CTIME
    SET CART03_CTIMEHH = F_CTIME_HH
    SET CART03_CTIMED1 = ':'
    SET CART03_CTIMEMM = F_CTIME_MM
    SET CART03_CTIMED2 = ':'
    SET CART03_CTIMESS = F_CTIME_SS
    SET LINE(/3/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART04_CDATE   = CDATE
    SET CART04_CUNIT   = CUNIT
    SET CART04_DEN     = DEN
    SET CART04_TRTCH   = TRTCH
    SET LINE(/4/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART05_LRECL   = LRECL
    SET CART05_BLKSIZE = BLKSIZE
    SET CART05_RECFM   = RECFM
    SET CART05_VOLSEQ  = VOLSEQ
    SET CART05_VOLCNT  = VOLUME_COUNT
    SET LINE(/5/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART06_BLKCNT  = BLKCNT
    SET CART06_FILESEQ = FILESEQ
    SET CART06_FORGN   = SCRATCH_SOURCE
    SET CART06_LABEL   = LABEL_P
```

```
   SET CART06_ERG     = PERM_WRITE_CLEAN
   SET LINE(/6/)      = WORK_LINE

   SET WORK_LINE      = ' '
   SET CART07_ACCT    = ACCT_P
   SET CART07_CPUID   = CPUID
   SET LINE(/7/)      = WORK_LINE

   PERFORM LABEL_OUT
 ENDPROC

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!      FORMAT AND OUTPUT A REEL TYPE LABEL                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 REEL_LABEL: PROC

   SET OUTPUT_LABEL   = ' '            15
   SET ROWS           = 10
   SET COLUMNS        = 80

   SET WORK_LINE      = ' '
   SET REEL01_DSN     = DSN
   SET LINE(/1/)      = WORK_LINE

   SET WORK_LINE      = ' '
   SET REEL02_CJOB    = CJOB
   SET REEL02_CSTEP   = CSTEP
   SET F_CTIME        = CTIME
   SET REEL02_CTIMEHH = F_CTIME_HH
   SET REEL02_CTIMED1 = ':'
   SET REEL02_CTIMEMM = F_CTIME_MM
   SET REEL02_CTIMED2 = ':'
   SET REEL02_CTIMESS = F_CTIME_SS
   SET REEL02_CDATE   = CDATE
   SET REEL02_CUNIT   = CUNIT
   SET REEL02_DEN     = DEN
   SET LINE(/2/)      = WORK_LINE

   SET WORK_LINE      = ' '
   SET REEL03_CPUID   = CPUID
   SET REEL03_LRECL   = LRECL
   SET REEL03_BLKSIZE = BLKSIZE
   SET REEL03_RECFM   = RECFM
   SET REEL03_BLKCNT  = BLKCNT
   SET REEL03_ERG     = PERM_WRITE_CLEAN
   SET REEL03_FILESEQ = FILESEQ
   SET LINE(/3/)      = WORK_LINE
```

```
    SET WORK_LINE      = ' '
    SET REEL04_VOLSER  = VOLSER
    SET REEL04_FORGN   = SCRATCH_SOURCE
    SET REEL04_VOLSEQ  = VOLSEQ
    SET REEL04_VOLCNT  = VOLUME_COUNT
    SET REEL04_LABEL   = LABEL_P
    SET REEL04_TRTCH   = TRTCH
    SET REEL04_ACCT    = ACCT_P
    SET LINE(/4/)      = WORK_LINE

    PERFORM LABEL_OUT

ENDPROC

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!      FORMAT AND OUTPUT CART TYPE SETUP LABELS                              !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 SETUP_CART: PROC                          16
    PERFORM SELECT_CART_OUTPUT

    SET OUTPUT_LABEL   = ' '
    SET ROWS           = 10
    SET COLUMNS        = 80

    SET WORK_LINE      = ' '
    SET CART01_DSN     = 'XXXXXXXXXXXXXXXXXXXXXXXXXX'
    SET LINE(/1/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART02_DSN2    = 'XXXXXXXXXXXXXXXXXX'
    SET CART02_CJOB    = 'XXXXXXXX'
    SET LINE(/2/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART03_VOLSER  = 'XXXXXX'
    SET CART03_CSTEP   = 'XXXXXXXX'
    SET CART03_CTIMEHH = 99
    SET CART03_CTIMED1 = ':'
    SET CART03_CTIMEMM = 99
    SET CART03_CTIMED2 = ':'
    SET CART03_CTIMESS = 99
    SET LINE(/3/)      = WORK_LINE
```

```
    SET WORK_LINE      = ' '
    SET CART04_CDATE   = 99999
    SET CART04_CUNIT   = 'XXXX'
    SET CART04_DEN     = 'XXXX'
    SET CART04_TRTCH   = 'XXXX'
    SET LINE(/4/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART05_LRECL   = 99999
    SET CART05_BLKSIZE = 999999
    SET CART05_RECFM   = 'XXXX'
    SET CART05_VOLSEQ  = 9999
    SET CART05_VOLCNT  = 999
    SET LINE(/5/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET CART06_BLKCNT  = 9999999999
    SET CART06_FILESEQ = 99999
    SET CART06_FORGN   = 'X'
    SET CART06_LABEL   = 'X'
    SET CART06_ERG     = 999
    SET LINE(/6/)      = WORK_LINE
    SET WORK_LINE      = ' '
    SET CART07_ACCT    = 'XXXXXXXXXXXX'
    SET CART07_CPUID   = 'XXXX'
    SET LINE(/7/)      = WORK_LINE

 SETUP_CART_LOOP:                          17
    PERFORM LABEL_OUT

    SET SETUP_COUNT = SETUP_COUNT + 1

    IF SETUP_COUNT < SETUP_MAXIMUM
       GOTO SETUP_CART_LOOP
    ENDIF
ENDPROC
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!      FORMAT AND OUTPUT REEL TYPE SETUP LABELS                          !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SETUP_REEL: PROC
    PERFORM SELECT_REEL_OUTPUT          18

    SET OUTPUT_LABEL   = ' '
    SET ROWS           = 10
    SET COLUMNS        = 80

    SET WORK_LINE      = ' '
    SET REEL01_DSN     = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    SET LINE(/1/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET REEL02_CJOB    = 'XXXXXXXX'
    SET REEL02_CSTEP   = 'XXXXXXXX'
    SET REEL02_CTIMEHH = 99
    SET REEL02_CTIMED1 = ':'
    SET REEL02_CTIMEMM = 99
    SET REEL02_CTIMED2 = ':'
    SET REEL02_CTIMESS = 99
    SET REEL02_CDATE   = 99999
    SET REEL02_CUNIT   = 'XXXX'
    SET REEL02_DEN     = 'XXXX'
    SET LINE(/2/)      = WORK_LINE

    SET WORK_LINE      = ' '
    SET REEL03_CPUID   = 'XXXX'
    SET REEL03_LRECL   = 99999
    SET REEL03_BLKSIZE = 999999
    SET REEL03_RECFM   = 'XXXX'
    SET REEL03_BLKCNT  = 9999999999
    SET REEL03_ERG     = 9999
    SET REEL03_FILESEQ = 99999
    SET LINE(/3/)      = WORK_LINE
    SET WORK_LINE      = ' '
    SET REEL04_VOLSER  = 'XXXXXX'
    SET REEL04_FORGN   = 'X'
    SET REEL04_VOLSEQ  = 9999
    SET REEL04_VOLCNT  = 999
    SET REEL04_LABEL   = 'X'
    SET REEL04_TRTCH   = 'XXXX'
    SET REEL04_ACCT    = 'XXXXXXXXXXXX'
    SET LINE(/4/)      = WORK_LINE
```

```
SETUP_REEL_LOOP:
    PERFORM LABEL_OUT         19

    SET SETUP_COUNT = SETUP_COUNT + 1

    IF SETUP_COUNT < SETUP_MAXIMUM
       GOTO SETUP_REEL_LOOP
    ENDIF
 ENDPROC

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! CALL CTSLBLOT WITH A LABEL REQUEST TO PRINT THE LABEL IMAGE         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
LABEL_OUT: PROC                                  20
    SET REQUEST = 'L'
    CALL CTSLBLOT USING OUTPUT_LABEL COMM_AREA
ENDPROC

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  SET THE DESTINATIONS FOR CARTS, REELS, AND LABEL IMAGES BELOW.     !
!                                                                     !
!  FOR PRT1_REQ, PRT2_REQ AND PRT3_REQ                                !
!      Y = PRINT LABEL, N = DON'T PRINT                               !
!                                                                     !
!  FOR WTO1_REQ, WTO2_REQ AND WTO3_REQ                                !
!      000 = DON'T PRINT, 1 - 128 IS WTO ROUTCDE TO PRINT             !
!                                                                     !
!  FOR CCI_DEST                                                       !
!      BLANK = DON'T SEND, NON-BLANK IS CCI ID                        !
!                                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!    SELECT WHERE CART LABELS WILL GO                                 !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SELECT_CART_OUTPUT: PROC                          21
     SET PRT1_REQ = 'Y'            !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     SET PRT2_REQ = 'N'            !! SEND LABEL IMAGES TO WTO     !!
     SET PRT3_REQ = 'N'            !! ROUTE CODES 13.              !!
     SET WTO1_REQ = 013            !!                              !!
     SET WTO2_REQ = 000            !! ALSO SEND COPY TO PRINTER1.  !!
     SET WTO3_REQ = 000            !!                              !!
     SET CCI_DEST = ' '            !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ENDPROC
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!    SELECT WHERE REEL LABELS WILL GO                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SELECT_REEL_OUTPUT: PROC                        21
     SET PRT1_REQ = 'Y'             !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     SET PRT2_REQ = 'N'             !! SEND LABEL IMAGES TO WTO     !!
     SET PRT3_REQ = 'N'             !! ROUTE CODES 14.              !!
     SET WTO1_REQ = 014             !!                              !!
     SET WTO2_REQ = 000             !! ALSO SEND COPY TO PRINTER1.  !!
     SET WTO3_REQ = 000             !!                              !!
     SET CCI_DEST = ' '             !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ENDPROC

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!    SELECT WHERE PREFORMATED LABELS WILL GO                              !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SELECT_IMAGE_OUTPUT: PROC                       21
     SET PRT1_REQ = 'Y'             !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     SET PRT2_REQ = 'N'             !! SEND LABEL IMAGES TO WTO     !!
     SET PRT3_REQ = 'N'             !! ROUTE CODES 13 AND 14.       !!
     SET WTO1_REQ = 013             !!                              !!
     SET WTO2_REQ = 014             !! ALSO SEND COPY TO PRINTER1.  !!
     SET WTO3_REQ = 000             !!                              !!
     SET CCI_DEST = ' '             !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ENDPROC
END
```

1

The first part of the code is a comment section showing the output format of labels produced by default. See Label formats for REELS and CARTS for additional information.

2

The COPY command brings in the EARL label and record definitions from members LABELDEF, CTEARLCM and CTEARLCN from CAI.CTAPECPB.

3

This section of code defines the storage array for the labels. The definition may be larger than the actual storage requirements.

4

The size for OUTPUT_LABEL is the maximum buffer size (10 rows X 80 columns +2).

5

The LINE array should be set to the number of rows used to calculate OUTPUT_LABEL.

For example

```
DEF OUTPUT_LABEL (X 1282) = '   ' 'OUTPUT' 'LABEL'
  .
  .
 DEF LINE ARRAY 16 = OUTPUT_LABEL 003-082 X
```

This defines a storage area that can accommodate labels up to 16 rows by 80 columns.

6

These fields are used to format fields for printing.

7

Defines the work lines used to build the label array. For ease in understanding the variable names in this section are defined as ttttnn-ffff where tttt indicates REEL or CART, nn is the two digit line number within the label and ffff describes the value to be printed.

The variables are defined as sub-strings of WORK_LINE at the positions indicated.

For example

The fourth line of the cartridge label displays the creating unit in columns 8-11 and the create date in columns 12-26.

8

The program allows you to produce setup labels if they are desired. The delivered sample automatically produces 10 setup labels in a CART format. If the REEL format is preferred, the number of setup labels needs to be changed or the setup labels are to be bypassed entirely, modify the default values in this section.

9

This section prints SETUP labels if requested. Checks are made to determine if SETUP labels are requested for CART, REEL, or BOTH.

10

This section gets the next label request. TLMSLBLS waits here until another label request is available.

11

This is a test for a termination request.

12

This section handles a SPINOFF request for one or more printers.

13

User code should be inserted here. You may bypass printing a label with a GOTO GETTAG or change the data being passed.

14

This section of the code formats labels for CARTS. Formatting is done one line at a time, and then moved to the label output area. Then the label output module is called.

15

This section of the code formats labels for REELS. Formatting is done one line at a time, and then moved to the label output area. Then the label output module is called.

16

This section of the code formats labels for CART SETUP labels. Formatting is done one line at a time, and then moved to the label output area. Then the label output module is called.

17

This section of the code formats labels for REEL SETUP labels. Formatting is done one line at a time, and then moved to the label output area. Then the label output module is called.

18

This section of the code causes the CART SETUP labels to be printed for the number of labels specified in SETUP_MAXIMUM.

19

This section of the code causes the REEL SETUP labels to be printed for the number of labels specified in SETUP_MAXIMUM.

20

This is the actual call to the LABEL output module.

21

These routines set the output destinations for the labels.

Accessing User Data

User data is 'free form' text that can be passed to your EARL label formatter. Realtime Auxiliary Disposition provides this capability when the user data option is specified. Up to 51 characters of free form text may be passed for each volume. The length of the user data is provided in the field LABEL_USERDATA_LEN, and the actual text is provided in the field LABEL_USERDATA.

Initialization Procedures

Input Requests

The input processor receives all input records and requests through the CA Common Services, CAICCI - Common Communicators Interface. These requests include the following:

C (Close)

This request is used to ask the Output processor to close all files

D (Data)

The common tape record is built based on the information available at the time of the request

L (Label)

The current record contains a preformatted label image ready to print

S (Spinoff)

This request closes and reopens any dynamically allocated printers

V (Volume)

A volume lookup is performed against the current tape database

The installed tape management system programmatically generates each label request or data record and sends the information to the label processor using the CA Common Services CAICCI - Common Communicators Interface.

Output Labels

The label output processor supports up to seven outputs for a single label image. Up to three WTO route codes may be used and up to three label printers. Additionally, a CAICCI destination can be used to redirect the label image to another label processor running on another CPU.

In this case, the CTS task and LAB subtask need to be started on the secondary CPU as well. Additionally, the CA Common Services CAICCI network must include both the primary and secondary CPU.

After this is in place, the client can direct the label image to the secondary CPU by defining the CAICCI destination as an output assignment on the label request within the EARL component.

CA Common Services CAICCI

CA Common Services CAICCI/CAIENF are used throughout the label processing. The following CAICCI node prefixes are used:

CA$FO...

This identifies the input label processor that receives all label requests.

CA$FS...

This identifies the originating sender of the label request.

CA$FW...

This identifies the output label processor when used to re-route a label image.

For more information on CA Common Services CAICCI, see the *CA Common Services Reference Guide*.

LAB Description

When LAB is started, it initiates the label processor and wait for either the label processor to complete or a command from the operator to be entered.

The label processor completes under any of the following conditions:

- The label input processor received a CLOSE request
- An error was detected in the EARL component
- An end-of-file condition was forced in the EARL component
- A fatal error occurred in either the input or output label processor
- A STOP or CANCEL command was passed to the LAB subtask

The CLOSE request shuts down the entire label process in a normal fashion. CA Common Services CAICCI used in the input processor is terminated. The EARL component receives the CLOSE request which in turn is passed to the output label processor, requesting it to close all opened files and terminate any use of CA Common Services CAICCI.

The following command in the LAB subtask is used to send the CLOSE request to the label processor.

```
F CTS,STOP LAB
```

or

```
F CTS,MSG LAB,STOP
```

Additionally, if the EARL component detects an error, it forces an END-OF-FILE condition which in turn requests the label processor to shutdown. The same is true for any fatal errors that may occur.

Startup Procedure

The CTS startup procedure contains all JCL required to initiate the LAB subtask including the label processor and the EARL component.

LAB can be started manually or automatically when CTS is started. CA Common Services CAIENF must have started and CAICCI communications must be up and operational before LAB is started.

Automatic Startup

To automatically start the LAB subtask when CTS is started, add the following command to the member CTSSTART in CAI.CTAPOPTN:

```
START  LAB
```

After the member CTSSTART is updated to contain the command listed above, CTS automatically starts the LAB subtask when CTS is started.

Manual Startup

After the CTS task has been started, you can communicate with CTS by using a MODIFY command or by WTOR.

The following is an example of how to manually start the LAB subtask under CTS by using the MODIFY command:

```
F CTS,START LAB
```

LAB Commands

Commands are sent to LAB using the CTS Message command (MSG). The following format should be used on all commands issued to the LAB subtask:

```
MSG LAB,xxxxxxxxxx
```

Where xxxxxxxxxx is the LAB command and must contain one of the following values:

STOP

to terminate label processing in a normal fashion

CANCEL

to cancel LAB including label processing with a User-111 Abend

STATUS

to request a current label processing status

VOLUME xxxxxx

to send a VOLUME request through the CA Common Services CAICCI to the label processor.

SPINOFF |xxxxxxxx

to send a SPINOFF request through CA Common Services CAICCI to the label processor.

The STOP command issues a CLOSE request to the label processor which terminates the process in a normal fashion. Once termination is complete, the LAB subtask ends.

The STATUS command issues a request that provides two messages indicating the time and date the label processor started along with accumulated totals of VOLUME, DATA, LABEL and SPINOFF CAICCI requests that have been received. Additionally, an error count associated with VOLUME requests is provided.

The VOLUME command directs a request for a specific volume serial number to be processed. This command requires a parameter containing the volume serial number associated with the request. For this reason, the command and parameter entered must be separated by a blank and enclosed within single quotes or parentheses, for example, 'VOLUME 980010'.

The SPINOFF command can be entered with or without a parameter.

If the SPINOFF command is entered without a parameter, the label processor closes, reallocates, and reopens all printer files that were originally dynamically allocated as a SYSOUT file, thus freeing the current SYSOUT to be printed.

If the SPINOFF command is entered with a parameter, the parameter must be either PRINTER1, PRINTER2 or PRINTER3 indicating the specific printer to which the request applies. Additionally, only the printer specified is closed, reallocated and reopened, freeing the current SYSOUT to be printed. In this case, the command and parameter entered must be separated by a blank and enclosed within single quotes or parentheses, for example, 'SPINOFF PRINTER1'.

Output Assignments

Up to seven outputs can be requested on each label image. Three different printers can be requested (PRINTER1, PRINTER2 and PRINTER3), along with up to three unique WTO Route Codes and additionally a CAICCI destination can be requested.

Unless the CTS started task procedure in CAI.CTAPPROC is modified, the particular printer is dynamically allocated as a SYSOUT using the OUTPUT statements OUTPUTP1, OUTPUTP2, OUTPUTP3 for printers PRINTER1, PRINTER2 and PRINTER3 respectively.

If a subsequent SPINOFF request is processed, all printers that were initially dynamically allocated are closed (and freed), reallocated and reopened. This frees the current SYSOUT to be printed.

Fields: PRT1_REQ, PRT2_REQ and PRT3_REQ

Representing requests for the PRINTER1, PRINTER2 and PRINTER3 files respectively, these fields may contain a value of *Y* (to request that the label be output to the respective printer) or *N* (to indicate that the respective printer is not to be used)

Fields: WTO1_REQ, WTO2_REQ and WTO3_REQ

These fields can be used to request up to three WTO route codes to be used to print the label image. Valid values for these fields are 000 through 128. Zero in the option indicates that the specific WTO route code is not to be used for the request.

It should be noted that using WTO route code 011 may conflict with warning and error messages issued by the input and output processors and is discouraged for this reason.

Field: CCI_DEST

If you wish to generate the labels on one CPU and route the label image to another CPU to print, you need to use the CAICCI Destination assignment.

If this is the case, you need to have two LAB subtasks running under CTS; one on the sending CPU and one on the receiving CPU. Additionally, the two CPUs need to be on the same CAIENF/CAICCI network in order to facilitate the cross-CPU communications. See the *CA Common Services Reference Guide* for more information on establishing CAIENF/CAICCI networks.

A CAICCI destination code may be entered if you wish the output processor to route the label image to another common tape labels started task on another CPU. The value entered in this field must contain CA$FO immediately followed by the four-character CPU identification where the receiving Common Tape labels started task is executing. The output processor validates the CAICCI Identification entered is up and operational before sending the label request through CAICCI Communications to the secondary label processor. No output assignments are propagated on label requests sent using CAICCI services. Any existing assignments can be modified by using the SET statement.

The ORIG_ID field contains the originating CAICCI Identification of the current request. SOURCE_ID contains a one-position value to identify the source of the current request. Possible values are B (Batch) or O (Online). These two fields can be used by the EARL program to select labels for printing.

If none of these outputs are requested, PRINTER1 is used by the output processor to print all label images.

Note: The delivered sample (TLMSLBLS) does not establish any output options, so label images for both REEL and CART densities (setup and actual labels) default to the same output (PRINTER1). If you use both REEL and CART densities, you need to modify a copy of the sample to establish a different output for each type of label being generated.

Sample Labels

The following pages describe the sample labels that are delivered for your use. These samples illustrate a few of the ways you can generate labels using the input and output processors through EARL along with the Online Label Interface.

Included in the sample is a section that automatically produces setup labels for printer alignment. The sample prints 10 setup labels using either the CART or REEL format. This section can be bypassed altogether with a minor change to the sample. Additionally, the section defaults to using the CART format and the default printer assignment. If this or other modifications to the section are required, you may want to modify the sample prior to using it.

Since these samples probably do not conform to your shop standards, you should copy these for your own use and make the necessary modifications to your own version.

Label formats for REELS and CARTS

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! 00000010

!  SAMPLE OUTPUT LABELS FOR REELS AND CARTRIDGES                        ! 00000020
!                                                                       ! 00000030
!  THE FOLLOWING FORMATS ARE BEING PRODUCED IN THIS SAMPLE:             ! 00000040
!                                                                       ! 00000050
!  REELS:                                                               ! 00000060
!     ....+....1....+....2....+....3....+....4....+                     ! 00000070
!                                                                       ! 00000130
!  1  DSN.....                                      ! 00000140
!  2  JOBNAME. STEPNAME HH:MM:SS YYDDD UUUU DEN.                        ! 00000150
!  3  CPU. LRECL BLKSI. RFM. BLKCNT.... ERG FSN...                      ! 00000160
!  4  VOLSER # VLSQ/VCNT T X       JOBACT.....                          ! 00000170
!                                                                       ! 00000180
!  CARTS:                                                               ! 00000190
!     ....+....1....+....2....+....                                     ! 00000200
!                                                                       ! 00000290
!  1  DSN.....                                                          ! 00000300
!  2   DSN2.............. JOBNAME.                                       ! 00000310
!  3   VOLSER STEPNAME HH:MM:SS                                        ! 00000320
!  4   YYDDD UUUU DEN. X                                               ! 00000330
!  5   LRECL BLKSI. RFM. VSEQ/VCNT                                     ! 00000340
!  6   BLKCNT.... FSN... # T ERG.                                       ! 00000350
!  7   JOBACCT.....   CPU.                                             ! 00000360
!                                                                       ! 00000370
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! 00000380
```

The above formats were extracted from the delivered sample for CART and REEL labels.

In the sample format, the field names occupy the location where the field values are placed. The CART format is used for cartridge densities and all others default to the REEL format.

Additionally, the REEL format occupies 4 rows, each containing 44 columns while the CART format occupies 7 rows with each containing 29 columns. In the delivered sample, one output label is generated for both formats with ROWS=10 and COLUMNS=80. This provides 3 blank rows between each label for REEL labels and 2 blank rows between each label for CART labels.

Sample REEL Setup Label Output

The following is a sample output of setup labels generated using the REEL label format.

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXX XXXXXXXX 99:99:99  99999 XXXX XXXX
XXXX 99999 999999 XXXX 9999999999     99999
XXXXXX X 9999 999 X XXXX    XXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXX XXXXXXXX 99:99:99  99999 XXXX XXXX
XXXX 99999 999999 XXXX 9999999999     99999
XXXXXX X 9999 999 X XXXX    XXXXXXXXXXXX
```

Sample REEL Label Output

The following is a sample output of labels generated that contained a REEL density.

```
 PAY.CHECK.AUDIT.COPY
 PAY108   STP01    14:50:02  97006 4082  200
 XE90 00080 032000 FB   0000005500 000 00002
 ODD003   0001 001 S         1-22-333-444
```

```
PAY.CHECK.BACKUP
PAY108   STP01    14:50:03  97006 0E82  200
XE90 00080 032000 FB   0000005500 000 00003
DD003   0001 001 S         1-22-333-444
```

Sample CART Setup Label Output

```
      XXXXXXXXXXXXXXXXXXXXXXXX
       XXXXXXXXXXXXXXXXXX XXXXXXXX
       XXXXXX XXXXXXXX  99:99:99
       99999 XXXX XXXX XXXX
       99999 999999 XXXX 9999 0999
       9999999999 99999  X X 0999
       XXXXXXXXXXX    XXXX
      XXXXXXXXXXXXXXXXXXXXXXXXX
```

```
      XXXXXXXXXXXXXXXXXX XXXXXXXX
       XXXXXX XXXXXXXX  99:99:99
       99999 XXXX XXXX XXXX
       99999 999999 XXXX 9999 0999
       9999999999 99999  X X 0999
       XXXXXXXXXXX    XXXX
```

Sample CART Label Output

```
PAY.CHECK.FICHE.TAPE
                    PAY108
 FICHE1 STP01     14:50:02
 97006 4E82 CART
 00080 032000 FB   0001 0000
 0000005500 00001    N 0000
 1-22-333-444    XE90
```

```
 PAY.CHECK.FICHE.TAPE
                     PAY108
 FICHE2 STP01     14:50:02
 97006 4E82 CART
 00080 032000 FB   0002 0002
 0000005500 00001    N 0000
 1-22-333-444    XE90
```

Main CA TLMS Processor (MAIN)

Before starting the TLMS subtask, be sure that TLMSRIM has been run. This is the Online Recorder task.

Startup Procedure

TLMS can be started manually or automatically when CTS is started. The recommended method is to have CTS automatically start the TLMS subtask.

Automatic Startup

To automatically start the TLMS subtask when CTS is started, add the following command to the member CTSSTART in the CAI.CTAPOPTN procedure.

```
START TLMS
```

After the member CTSSTART is updated to contain the command listed above, CTS automatically starts the TLMS subtask when CTS is started.

Manual Startup

After the CTS task has been started, you can communicate with CTS by using a MODIFY command or by WTOR.

The following is an example of how to manually start the TLMS subtask under CTS by using the MODIFY command:

```
F CTS,START TLMS
```

TLMS Commands

Commands are sent to TLMS using the CTS Message command (MSG).

The following format should be used on all commands issued to the TLMS subtask:

`MSG TLMS,`*`xxxxxxxxxx`*

xxxxxxxxxx

Specifies the TLMS command and must contain one of the following values:

STOP

To request TLMS to close all files and terminate normally.

CANCEL

To request TLMS to cancel itself with a User-111 Abend.

STATUS

To request TLMS to provide the current status.

TRACE|NOTRACE

TLMS produces a snap of each transaction before and after it is processed by TLMS.

Note: This replaced the TRACK function of OMS. The CTS option SNAP must be specified. The TRACE data may be printed by the command SPINOFF SNAP to CTS.

Inquiry Update Task (INQR)

The INQR task allows online inquiry and update through an outstanding WTOR. Output goes to the console specified by the ROUTINQ option and is also recorded to the CTS log. This is the same program (TLMSINQR) which can execute as a batch job or as an OS/390 started task. CA TLMS INQR automatically detects that it is a CTS subtask, and can receive commands through the CTS message facilities.

Startup Procedure

INQR can be started manually or automatically when CTS is started. The recommended method is to have CTS automatically start the INQR subtask.

Automatic Startup

To automatically start the INQR subtask when CTS is started, add the following command to the member CTSSTART in the CAI.CTAPOPTN procedure.

```
START INQR
```

After the member CTSSTART is updated to contain the command listed above, CTS automatically starts the INQR subtask when CTS is started.

Manual Startup

After the CTS task has been started, you can communicate with CTS by using a MODIFY command or by WTOR.

The following is an example of how to manually start the INQR subtask under CTS by using the MODIFY command:

```
F CTS,START INQR
```

Distributed Tape Support (DTS)

Distributed Tape Support (DTS) provides a mechanism for recording the tape inventories of distributed backup servers in the VMF. This gives you a complete picture of all tapes in your entire enterprise, both mainframe and distributed. Depending on the distributed backup product in use, varying degrees of information is collected. This information would include the volume serial number, creation date, and other data relative to tape media. The following distributed backup products are currently supported:

- CA ARCserve Backup for Windows r11
- CA ARCserve Backup for UNIX r11
- Veritas NetBackup for Windows
- Tivoli Storage Manager for Windows

This service runs in the CTS address space by the program named CTSDTS under the service name of DTS.

iSponsor Technology

CTSDTS makes use of CA iSponsor/iGateway technology developed for the BrightStor Portal product to retrieve the data from the remote backup servers over TCP/IP. iSponsors are hosted by the iGateway that resides on the target server. The iGateway is a layer of software that communicates with the IP stack to receive and respond to requests for iSponsors.

The iSponsors themselves are applets consisting of multiple "methods" that will extract data from the server and transmit this data over the iGateway to the requestor. Each "method" belonging to a particular iSponsor provides a unique view of the backup server's database.

For each supported distributed backup product, there is at least one iSponsor. For certain supported products, there are multiple iSponsors, one for each supported release. In order to retrieve data from a remote backup server, two components must be installed on each remote server to be accessed: the iGateway and the appropriate iSponsor. iSponsors and iGateway components may be downloaded free of charge from :http//ca.com/support. Refer to the Product Home Page for BrightStor Portal for further information on iSponsor/iGateway technology.

Method of Operation

The target distributed servers are defined to DTS using control statements. When a remote server's inventory is to be queried, DTS dispatches a query task to establish a session with the target server. DTS identifies what iSponsors are running on a remote server and selects the appropriate iSponsor and method. DTS uses the selected method to query the backup server's database. The retrieved data is then used to construct a Distributed Tape Inventory File. The current inventory is compared to the previous inventory.

DTS then updates the VMF to reflect any changes detected in that server's inventory. This validation is done in order to reduce redundant updates. DTS logs query activity to a sysout file. The level of logging is controlled with the MSGLEVEL parameter.

Execution of data retrieval sessions can be triggered periodically using the scheduling component of CTS, CTSSCHD. Data retrieval sessions can also be triggered manually using a console command.

Requirements

Volume Serial Number Ranges in the VMF

Volume serial ranges that are retrieved from the remote backup servers must be defined to CA TLMS and must be marked as AGENT tapes. Use the following command to mark tapes:

UPV *volser* AGENT=Y

volser

The volume serial number to be updated.

Use the ranges for distributed systems only and do not share with mainframe applications. Ensure that the same volume serial number ranges are not used by different distributed backup systems.

Note: DTS tapes are also referred to as Agent tapes.

CTS Proc Changes

A SYSTCPD DD statement is required for TCP/IP services. Refer to your TCP/IP installation for the correct dataset name.

A CTSOPTNS DD statement is required to access the CAI.CTAPOPTN dataset. This dataset contains the parameters for the DTS feature. The sample CTS procedure contains this DD as a comment. You can uncomment this DD, or supply your own. The format is:

```
//CTSOPTNS DD DISP=SHR,DSN=CAI.CTAPOPTN
```

Security Requirements

The DTS feature dynamically creates MVS datasets for temporarily storing remote server data. The dataset name is controlled by the PREFIX= operand on the GLOBAL statement appended with the SRVDEF name itself. You may need to update your security system to provide the CTS started task appropriate authority to create these datasets. Verify with your security administrator that the USERID assigned to the CTS started task has authority to create these MVS data sets.

The DTS feature makes use of TCP/IP for communicating with the remote servers. The USERID associated with the CTS started task must have a valid OMVS segment defined to allow use of TCP/IP facilities. The USERID assigned to your TCP/IP started task may be used as a model to construct a security profile for the CTS started task. Consult with your mainframe security administrator for this requirement.

Startup Procedure

DTS can be started up manually when CTS is started. The recommended method is to have CTS automatically start the DTS subtask.

Note: Starting the DTS subtask does not automatically initiate communication with the backup servers defined. Scheduling regular communication with the backup servers is accomplished using the CTSSCHD subtask.

Automatic Startup

To automatically start the DTS subtask each time CTS is started, add the following commands to the CTSSTART member in CAI.CTAPOPTN:

```
SET TASK(DTS) PGM(CTSDTS) PARM(xx)
START DTS
```

xx is the suffix for the CTODTSxx member to use. The default is 00 if not specified.

Manual Startup

After the CTS task has been started, you can communicate with CTS by using a MODIFY command or by replying to the outstanding CTS WTOR.

The following is an example of how to manually start the DTS subtask under CTS by using the modify command:

```
F CTS,SET TASK(DTS) PGM(CTSDTS) PARM(XX)
F CTS,START DTS
```

XX is the suffix for the CTODTSxx member to use. The default is 00 if not specified.

DTS Control Statements Contained in CTODTS00

Control Member Name

DTS will read a member from the CAI.CTAPOPTN library on startup to define the remote servers and establish certain global defaults. The default name of the member is CTODTS00. The last two characters may be changed, and the PARM parameter on the START command can be changed to reflect the new member name.

Control Statements

Selectable DTS defaults are established with the GLOBAL control statement. Remote server definitions are established with the SRVDEF control statement. All control statement keywords and operands must be uppercase.

GLOBAL Control Statement

The GLOBAL control statement is coded as follows:

```
GLOBAL    TCPNAME=tcpipname,
[MAXTASK=nn,]
[PREFIX=perfixname,]
[UNIT=unitname,]
[VOL=volser,]
[MSGLEVEL=nn,]
[STORCLAS=storageclassname,]
[DATACLAS=dataclassname,]
[MGMTCLAS=managementclassname,]
[TRACE/NOTRACE]
```

TCPNAME=

A required parameter that specifies the name of the TCPIP address space on the mainframe where CTS is running.

MAXTASK=

An optional parameter from 1 to 10 that specifies the maximum number of concurrent data retrieval operations that can be active at any one time. This specification is used to throttle resource utilization when multiple servers require data retrieval at the same time. All data retrieval sessions are queued up and serviced by the available data retrieval subtasks.

UNIT=

An optional parameter that specifies a valid unitname to be used to dynamically allocate the distributed tape inventory file.

VOL=

An optional parameter that specifies the volume serial number to use when allocating the distributed tape inventory file.

MSGLEVEL=

An optional parameter that specifies the level of messaging that will occur. Valid values are 0 through 9. The default is 0 which will list only the most important messages. This can be changed with a console command during execution. Setting this to 9 will provide supplementary information that can be used to diagnose abnormalities.

STORCLAS=

An optional parameter that specifies a valid SMS storage class to be used when allocating the distributed tape inventory file.

DATACLAS=

An optional parameter that specifies a valid SMS data class to be used when allocating the distributed tape inventory file.

MGMTCLAS=

An optional parameter that specifies a valid SMS management class to be used when allocating the distributed tape inventory file.

TRACE/NOTRACE

An optional parameter that will display all iSponsors found on a remote backup server when DTS connects to it.

SRVDEF Control Statement

The SRVDEF control statement is coded as follows:

```
SRVDEF     NAME=srvdefname,
SERVER=servername,
[SCHD=schdname,]
[LOG/NOLOG]
[TRACE/NOTRACE]
```

NAME=

Any 1 to 8 character name for this SRVDEF entry. The first character must be alphabetic. This name is used as the lowest level qualifier in constructing the data set name of the Distributed Tape Inventory File for this server. DTS will create these files as needed.

SERVER=

The DNS name or IP address of the remote server running the backup product you wish to manage.

SCHD=

The name of the CTSSCHD schedule event which is to trigger a query session for this server. For example, if SCHD=HOURLY is specified, you must ensure the SCHD component of CTS is started and contains a definition for HOURLY.

LOG/NOLOG

An optional parameter specifying whether print logging of received data is desired. If set to LOG, a SYSOUT DD with the name of this SRVDEF will be dynamically allocated. NOLOG is the default.

TRACE/NOTRACE

An optional parameter that will display all iSponsors found on a remote backup server when DTS connects to it. If this parameter is specified, it will override the specification in the GLOBAL specification.

Operator Commands

DTS can receive commands from the console using the modify command, or from the outstanding CTS WTOR. The commands are as follows:

Start Command

```
F CTS,START,DTS[,PARM(xx)]
```

This command will start the DTS task. If you have specified DTS to be started in the CTS startup options, this command is not required. The optional parm parameter is used to specify a different suffix for the CTODTS member to read.

```
Stop Command
```

```
F CTS,STOP,DTS
```

This command will stop the DTS task. Any SRVDEF definitions that are currently being processed will be allowed to complete.

MSGLEVEL Command

```
F CTS,MSG,DTS,'MSGLEVEL=n'
```

This command will change the level of messaging of the DTS task. Valid values of *n* are 0 through 9, where 0 indicates that only critical error messages are to be displayed up to 9, which requests all detailed messages be displayed.

Start Processing Command

```
F CTS,MSG,DTS,'START NAME=srvdefname'
```

This command will start a data query session for the SRVDEF entry srvdefname.

Start a Group of SRVDEFs

```
F CTS,MSG,DTS,'EVENT SCHD=schdname'
```

This command will start a data retrieval session for all SRVDEF definitions that have specified schname as an operand of the SCHD parameter.

Start or Stop Displaying All Discovered iSponsors

```
F CTS,MSG,DTS,TRACE or NOTRACE
```

This command will initiate or suspend the display of iSponsors found on all SRVDEF definitions when a data retrieval session is started.

List the DEfined SRVDEF Definitions

```
F CTS,MSG,DTS,LIST
```

This command will produce a list of all SRVDEF definitions defined. The list will be written to the DTSLIST dd statement.

Turn Logging On or Off for a SRVDEF Definition

```
F CTS,MSG,DTS,[LOG][NOLOG],NAME=srvdefname
```

This command will turn logging on for the SRVDEF definition srvdefname. If you wish to suspend logging for this SRVDEF, specify NOLOG in place of LOG.

Sample SRVDEF Definitions

This is a sample defining one server. The server name is server1.yourdomain.com.

```
/*                                                    /*
/*                                                    /*
GLOBAL TCPNAME=TCPIP90,
       MAXTASK=2,
       UNIT=3390,
       PREFIX=CTS.ISPONSOR.DATA
SRVDEF NAME=SERVER1,
       SERVER=SERVER1.YOURDOMAIN.COM,
       SCHD=HOURLY
```

This is a sample defining one server using the ip address of the server. The server IP address is 138.42.10.105.

```
/*                                                     /*
/*                                                     /*
GLOBAL TCPNAME=TCPIP90,
       MAXTASK=2,
       UNIT=3390,
       PREFIX=CTS.ISPONSOR.DATA
SRVDEF NAME=SERVER1,
       SERVER=138.42.10.105,
       SCHD=HOURLY
```

CTS Scheduler (SCHD)

CTSSCHD is the scheduling component of CTS. It will send commands to any other CTS tasks at a pre-defined time. Schedules are defined using control statements. Each TOD element defined will begin with the "SCHD" verb and must have an eight character name, and a time of day value. The day of the week may optionally be specified. The action to be taken when an event has expired is described with the "WHEN" statement. Multiple TOD elements may be grouped together and assigned a single name to minimize coding requirements. This is accomplished using the GROUP operand in place of the TOD operand.

CTS Proc Changes

A CTSOPTNS DD statement is required to access the CAI.CTAPOPTN dataset. This dataset contains the parameters for the SCHD feature. The sample CTS procedure contains this DD as a comment. You can uncomment this DD, or supply your own. The format is:

```
//CTSOPTNS DD DISP=SHR,DSN=CAI.CTAPOPTN
```

Startup

Automatic Startup

To automatically start the SCHD subtask each time CTS is started, add the following commands to the CTSSTART member in CAI.CTAPOPTN:

```
SET TASK(SCHD) PGM(CTSSCHD) PARM(XX)
START SCHD
```

Where *XX* is the suffix for the CTOSCH member to use. The default is 00 if not specified.

Manual Startup

After the CTS task has been started, you can communicate with CTS by using a MODIFY command for by WTOR.

The following is an example of how to manually start the SCHD subtask under CTS by using the modify command:

```
F CTS,SET TASK(SCHD) PGM(CTSSCHD) PARM(XX)
F CTS,START SCHD
```

Where *XX* is the suffix for the CTOSCH member to use. The default is 00 if not specified.

SCHD Control Statements Contained in CTOSCH00

Control Member Name

SCHD will read a member from the CAI.CTAPOPTN library on startup to define schedules, groups, and actions. The default name of the member is CTOSCH00. The last two characters may be changed, and the PARM parameter on the START command can be changed to reflect the new member name.

Event Definition Statements

TOD events, event groups, and associated actions are defined in the CTOSCH00 member of CAI.CTAPOPTN. All control statements and keywords must be uppercase. The control statements are as follows:

TOD Event Definition

A TOD event is defined using the SCHD statement as follows:

```
SCHD     NAME=event.name,
TOD=hh.mm,
[DOW=(MON,TUE,WED,THU,FRI,SAT,SUN)]
```

NAME=

A required parameter that names this SCHD definition. **event.name** can be any unique alphanumeric character string up to 24 characters in length. The first character must be alphabetic.

TOD=

This parameter specifies the time of day this event is to occur. It is specified as hh.mm. Valid values are 00.00 through 23.59.

DOW=

This optional parameter specifies the day of the week this event should occur. Valid operands are MON,TUE,WED,THU,FRI,SAT,SUN. If multiple days are specified, they must be enclosed in parentheses. If omitted, the TOD event will occur once each day.

Group Definition

TOD events defined with the SCHD command may be grouped together under a single name. This allows coding a single action (WHEN) command that will be issued when any of the TOD events within the group occur. To group a set of TOD events, the SCHD command is used with the GROUP operand:

```
SCHD     NAME=group.name,
GROUP=(event.name,event.name,...)
```

NAME=

A required parameter that names this SCHD definition. **group.name** can be any unique alphanumeric character string up to 24 characters in length. The first character must be alphabetic.

GROUP=

This parameter is used to group multiple event definitions together and assign them a single name. The event names must be enclosed in parentheses. There is a limit of eight event names that can be specified for the GROUP operand. If more than eight event names are required to make up a group, use another SCHD statement with the same NAME operand as this one and specify the additional event entries. This process may be repeated as many times as necessary.

Action Statement Definition

When a discrete event occurs, or an event that is a member of a group, an action for this occurrence is triggered. This action will notify the target CTS task of the event. A default notification template will be used unless the MSG parameter is coded. The syntax of the WHEN statement is as follows:

```
WHEN(name) NOTIFY(cts.taskname) [MSG=
'message to send']
```

WHEN

The WHEN keyword specifies a previously defined TOD event or GROUP.

NOTIFY

This parameter is required. **cts.taskname** specifies the name of the CTS task to notify that the event has occurred. The task name is the same name defined in a SET TASK(...) initialization statement.

MSG

This optional parameter specifies the text to be sent to the specific CTS task when this event occurs. If not specified, the default message 'EVENT SCHD=**name**' will be sent to the CTS task where **name** is replaced with the name on the WHEN statement.

Sample Schedules

A Simple Schedule with One TOD Event and One Action Statement

```
/*                                                              */
/* The following statements describe one event at 08:00        */
/* and an action statement that will notify the DTS subtask.   */
/*                                                              */
SCHD NAME(EIGHTAM),TOD(08.00)
/*                                                              */
WHEN(EIGHTAM) NOTIFY(DTS)
/*                                                              */
/* End of sample                                               */
/*                                                              */
```

A Simple Schedule with Two TOD Events and Two Action Statements

```
/*                                                              */
/* The following statements describe one event at 08:00        */
/* and an action statement that will notify the DTS subtask.   */
/*                                                              */
SCHD NAME(EIGHTAM), TOD(08.00)
SCHD NAME(TENAM), TOD(10.00)
/*                                                              */
/*                                                              */
/* At 8 am, notify DTS.                                        */
/* At 10 am, notify the APEC task to start a scan.             */
/*                                                              */
WHEN(EIGHTAM) NOTIFY(DTS)
WHEN(TENAM) NOTIFY(APEC) MSG=SCAN
/*                                                              */
/* End of sample                                               */
/*                                                              */
```

A Schedule with Multiple TOD Events, a Grouping Entry, and One Action Statement

```
/*                                                                  */
/* The following statements define TOD events for hours 09          */
/* through 15, a grouping statement, and an action statement.*/
SCHD NAME(NINEAM) ,TOD(09.00)
SCHD NAME(TENAM) ,TOD(10.00)
SCHD NAME(ELEVENAM) ,TOD(11.00)
SCHD NAME(NOON) ,TOD(12.00)
SCHD NAME(ONEPM) ,TOD(13.00)
SCHD NAME(TWOPM) ,TOD(14.00)
SCHD NAME(THREEPM) ,TOD(15.00)
SCHD NAME(FOURPM) ,TOD(16.00)
/*                                                                  */
/*                                                                  */
/* Group all of the above TOD definitions to a single name,         */
/* "DAY_SHIFT"                                                      */
/*                                                                  */
/*                                                                  */
SCHD NAME(DAY_SHIFT),
     GROUP(NINEAM,TENAM,ELEVENAM,NOON,ONEPM,TWOPM,THREEPM,FOURPM)
/*                                                                  */
/*                                                                  */
/* The next statement will send the text "EVENT SCHD(DAY_SHIFT)"
/* to cts task DTS.                                                 */
/*                                                                  */
/*                                                                  */
WHEN(DAY_SHIFT) NOTIFY(DTS)
/*                                                                  */
/*                                                                  */
/* End of sample                                                    */
/*                                                                  */
/*                                                                  */
/*                                                                  */
```

A Schedule that Defines Every Hour of the Day, Multiple Grouping Statements, and Several Action Statements

Note that only one action statement is enabled. TOD specifications and group operands may not be continued. The group specifications are cumulative. You may repeat the group name when grouping multiple TOD events into a single group, as shown below.

```
/*                                                          */
/* The following statements describe events on each hour. */
/* The name of each event corresponds with the hour        */
/*                                                          */
SCHD NAME(ONEAM) ,TOD(01.00)
SCHD NAME(TWOAM) ,TOD(02.00)
SCHD NAME(THREEAM) ,TOD(03.00)
SCHD NAME(FOURAM) ,TOD(04.00)
SCHD NAME(FIVEAM) ,TOD(05.00)
SCHD NAME(SIXAM) ,TOD(06.00)
SCHD NAME(SEVENAM) ,TOD(07.00)
SCHD NAME(EIGHTAM) ,TOD(08.00)
SCHD NAME(NINEAM) ,TOD(09.00)
SCHD NAME(TENAM) ,TOD(10.00)
SCHD NAME(ELEVENAM) ,TOD(11.00)
SCHD NAME(NOON) ,TOD(12.00)
SCHD NAME(ONEPM) ,TOD(13.00)
SCHD NAME(TWOPM) ,TOD(14.00)
SCHD NAME(THREEPM) ,TOD(15.00)
SCHD NAME(FOURPM) ,TOD(16.00)
SCHD NAME(FIVEPM) ,TOD(17.00)
SCHD NAME(SIXPM) ,TOD(18.00)
SCHD NAME(SEVENPM) ,TOD(19.00)
SCHD NAME(EIGHTPM) ,TOD(20.00)
SCHD NAME(NINEPM) ,TOD(21.00)
SCHD NAME(TENPM) ,TOD(22.00)
SCHD NAME(ELEVENPM) ,TOD(23.00)
SCHD NAME(MIDNIGHT) ,TOD(00.00)
```

```
/*                                                                 */
/* The following statements group the events into groups.          */
/* The first group is the night_shift group, with entries t1       */
/* through 8.                                                      */
/*                                                                 */
SCHD NAME(NIGHT_SHIFT)
 GROUP(ONEAM,TWOAM,THREEAM,FOURAM,FIVEAM,SIXAM,SEVENAM,EIGHTAM)
/*                                                                 */
/* The next group, day_shift, describes the hours 9 t0 16.         */
/*                                                                 */
SCHD NAME(DAY_SHIFT)
   GROUP(NINEAM,TENAM,ELEVENAM,NOON,ONEPM,TWOPM,THREEPM,FOURPM)
/*                                                                 */
/* Finally, the evening_shift group describes hours 17 through*/
/* 24. The "EVENING_SHIFT" name is repeated in order to            */
/* specify the additional entries.                                 */
/*                                                                 */
SCHD NAME(EVENING_SHIFT)
     GROUP(FIVEPM,SIXPM,SEVENPM,EIGHTPM)
SCHD NAME(EVENING_SHIFT)
     GROUP(NINEPM,TENPM,ELEVENPM,MIDNIGHT)
/*                                                                 */
/* The next statements describe a group that has all hourly        */
/* entries in it. You can only specify 8 entries on the            */
/* group statement, so the entire hourly SCHD NAME(HOURLY)         */
/* statement is repeated, each having the next 8 hourly            */
/* entries in it.                                                  */
/*                                                                 */
```

```
SCHD NAME(HOURLY)
  GROUP(ONEAM,TWOAM,THREEAM,FOURAM,FIVEAM,SIXAM,SEVENAM,EIGHTAM)
/*                                                               */
/* Continue specifying the hourly group with the next 8 hours */
/* in it.                                                        */
/*                                                               */
SCHD NAME(HOURLY)
  GROUP(NINEAM,TENAM,ELEVENAM,NOON,ONEPM,TWOPM,THREEPM,FOURPM)
/*                                                               */
/* Continue specifying the hourly group with the next 8 hours */
/* in it.                                                        */
SCHD NAME(HOURLY)
     GROUP(FIVEPM,SIXPM,SEVENPM,EIGHTPM)
SCHD NAME(HOURLY)
     GROUP(NINEPM,TENPM,ELEVENPM,MIDNIGHT)
/*                                                               */
/* The next statement will send the message: EVENT SCHD(HRLY) */
/* to CTS task DTS. Since each TOD event is defined to HOURLY,*/
/* a message will be sent at the start of each hour.          */
/*                                                               */
/*                                                               */
WHEN(HOURLY) NOTIFY(DTS)
/*                                                               */
/*                                                               */
/* The format of a command to send a message for each event   */
/* in the day shift would be:                                    */
/*                                                               */


WHEN(DAY_SHIFT) NOTIFY(DTS)
/*                                                               */
/* The format of a command to send a message for each event   */
/* in the night shift would be:                                  */
/*                                                               */
WHEN(NIGHT_SHIFT) NOTIFY(DTS)
/*                                                               */
/* The format of a command to send a message for each event   */
/* in the evening shift would be:
/*                                                               */
WHEN(EVENING_SHIFT) NOTIFY(DTS)
/*                                                               */
/*                                                               */
/* End of sample                                                 */
/*                                                               */
```

CA TLMS Health Checker Service (HCK)

The CTSHCK subtask of the Common Tape System provides support for the CA TLMS health checks that run under the IBM Health Checker for z/OS to identify potential CA TLMS and other problems in your z/OS environment and to make recommendations on how to fix these potential problems. This subtask is always started when the CTS address space starts. All available CA TLMS health checks will be automatically discovered and started.

Startup Procedure

The HCK subtask is always started when the CTS address space starts. No control statements are necessary to run this subtask.

Operator Commands

REFRESH Command

```
F CTS,MSG,HCK,REINIT
```

This command will request that HCK stop and restart all available CA TLMS health checks. After applying CA TLMS maintenance that introduces new CA TLMS health checks this command can be used to cause the new health checks to be started. You do not have to stop and start the entire CTS address space.

START Command

```
F CTS,START,HCK
```

This command starts the HCK subtask. The HCK subtask is always started when the CTS address space initializes, so the only time this command should be needed is after a STOP command has been issued and you want health check processing to resume.

STOP Command

```
F CTS,STOP,HCK
```

This command stops the HCK subtask. No further CA TLMS health check processing will be performed until either the CTS task is restarted or the START command is issued.

Chapter 4: User Exits and Macros

CA TLMS provides several user exits and the TLMDATE macros that let you tailor the supplied code to meet your data center requirements.

This section contains the following topics:

The various forms of the TLMDATE macro let you perform date conversion and calculation routines using CA TLMS keywords, and internal and external dates in varying formats.

Note: For information on the TLMDATE macro, see TLMDATE Macro - Common Date Processing Routines (see page 347).

Installing User Exits

All CA TLMS user exits are supplied as source members in CAI.CTAPSAMP. If you need to code your own exit, you can write a simple SMP USERMOD using sample jobs provided for each exit. These jobs will apply the source update to the supplied source and use SMP to automatically assemble and link the affected member. (See sample members in the CAI.CTAPJCL library.) These USERMODs should only be RECEIVED and APPLIED, never ACCEPTED.

Using the TLMXBGN Macro for User Exits

TLMXBGN is a supplied macro for use in generating user exits for CA TLMS. Macro TLMXBGN ensures the current environment and provides the following standard functions for the systems programmer responsible for coding CA TLMS user exits:

- Generates entry and exit logic
- Establishes addressability and base registers
- Performs save area logic
- Equates registers
- Names CSECTs and provides "eye-catchers"
- GETMAINs save area and work area for the exit

Parameters which are supplied to the TLMXBGN macro are:

name TLMXBGN BASE=,EXIT=,LIST,RETURN=,SA=,WS=,SP=

name

Begins in position 1 and is the exit name, such as TLMSXC.

TLMXBGN

Begins in position 10 and identifies this macro.

BASE

Identifies the program base registers (2-12). Up to four are supported.

EXIT

Is the label for the R15 return code exit (user loads R15).

LIST

If specified, macro options are listed inline.

LS.

RETURN

Is the label for the zero return code exit.

SA

Is the label for the save area and exit work area DSECT.

SP

Is the subpool number of GETMAIN for WS.

Default: zero

WS

Is the size of GETMAIN for working storage. The default is 72 bytes for save area, and is *always* first. Register 13 is the first base register for the work area.

TLMSXCLS - CLOSE/EOV Exit

The CLOSE function of the CA TLMS SVC allows you to access its 408-byte transaction record through a user-written exit routine so that you may change or add data to the record. The data can be examined later by the online recorder user exit, TLMSXUPD. TLMSXCLS is taken from the CLOSE/EOV function of the module TLMSCLSE.

The assembler DSECT for the transaction record is named TRANREC, which maps message types 1 and 2.

This exit is activated by specifying YES or *exitname* in the CLSEXIT system option.

Use

The user exit, as supplied, performs no function, and will require the insertion of your own code to view or alter CLOSE tape transactions to CA TLMS. Care should be taken when determining which fields, if any, are to be altered. The transaction user data cell is 59 bytes and contains the only fields that would normally be changed.

Link edit this user exit as TLMSXCLS in the CA TLMS target load library.

Register Usage

R0

work register

R1

address of TRANREC

R2

address of 3-word list - TCB, SVRB, ASCB

R3

free

R4

free

R5

free

R6

free

R7

address of CLOSE DCB

R8

free

R9

free

R10

free

R11

address of options table TLMSIPO

R12

base register of exit

R13

exit save and work area

R14

return address

R15

work register

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU011).

This exit is activated by specifying YES for the CLSEXIT system option.

TLMSXCMD - Command Exit

TLMSXCMD allows you to access a 120-byte input command message area from various processing functions within CA TLMS. You may change keywords and restrict command functions processed by TLMSCMND. The call to this exit occurs before the call to CA TLMS security.

This exit receives control for all update and display commands from the batch inquiry/update function, TLMSINQR, and the TSO line-by-line command processor, TLMSTSO. The full-screen interface, TLTPISPF, only passes update commands to this exit.

The assembler DSECT for the input command area is named CMNDAREA; for the user message, it is named CMNDMSG. The user exit must be an executable load module in the CA TLMS target load library and must be named TLMSXCMD. If you are a TSO user, the CA TLMS load library must be accessible.

Parameter List

TLMSCMND passes buffer addresses to the exit routine using standard linkage conventions (register 1 points to a parameter list). The two buffer addresses are

- the address of the 120-byte input message
- the address of the 70-byte user message

The exit must set the return code (register 15) to zeros to allow continuation or to nonzero to reject the input and display the message returned by the exit. This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU012).

This exit is activated by specifying YES for the CMDEXIT system option.

TLMSXNIT - Tape Initialization User Exit

This exit allows you to ACCEPT or REJECT tapes mounted for initialization. You may also ACCEPT the initialization and MODIFY the TMVENDOR and TMUSER fields and the VMF record or the tape labels.

This exit is loaded once at the beginning of TLMSNITT and is called once for each tape that is valid for initialization.

This exit is invoked before the CATLTN01 message is issued.

This exit is activated by specifying YES or *exitname* in the NITEXIT system option.

Register Usage

Standard linkages are used. Register 13 points to a save area, Register 14 is the return address, Register 15 points to the entry point and must be loaded with a RETURN CODE on exit from the user exit. Register 1 points to a parameter list.

DC A(UXCMDSCT)

Up to an 8-character length description:

- INTAPE
- OUTTAPE
- OLDTAPE
- NEWTAPE

Followed by a single character:

- Y - For parameter = test
- N - Test not specified in JCL parameter

Followed by a single character:

- Y - For LABELDD present
- N - For LABELDD missing

Followed by a four character UCB for the Tape Unit. For example, the data passed in the argument might look like the following:

```
UXCMDSCT    DC   CL8'INTAPE',C'Y'
```

DC A(DBRECORD)

DB record for the volume specified on the SER input parameter. If the OWNER input parameter was specified the VENDOR field will be overlaid with the value specified with the OWNER parameter.

DC A(TAPELABS)

80-character label records read by CTSTIO (for example, VOL1, HDR1, HDR2, UHL? labels, CL8'TAPEMARK' dummy record), ASCII characters in ANSI labels will have already been converted to EBCDIC. Following the C'TAPEMARK' dummy record is one of the following 3-character length label types:

NL

No labels (Operator interrogated for ID of mounted tape).

SL

Standard labels.

SUL

Standard and user labels.

AL

ANSI labels.

AUL

ANSI and user labels.

OPR

Reeled tape with command not equal to OUTTAPE (operator interrogated for ID).

???

Tape was written by an incompatible drive. However, the volume ID could be obtained from the sense data. The actual label records could not be read and thus, the type of labels on the tape cannot be determined (NEW VALUE).

(blank)

Blank tape (3480 or 3490 only).

For example, the data passed in the argument might look like the following:

```
LABELS  DC      CL80'VOL1...'
        DC      CL80'HDR1...'
        DC      CL80'HDR2...'
        DC      CL80'TAPEMARKSL'
```

DC A(LBRECORD)

DB record for volume mounted (before initialization). This always contains a value in VOLSER. When there is no record in the VMF for the volume, the remainder of the area is set to binary zeroes to indicate that this is a dummy record. In the event of a new blank tape, the VOLSER field will contain blanks. User exits should use this field to obtain the identity of the mounted volumes, since a VOL1 record may not always be present in the LABEL's argument.

DC A(X'8000000'+UXOWSCT)

Owner data to be written to the tape, value of OWNER parameter if specified, otherwise from the DBRECORD for the SER specified in the input (2nd argument).

Return Code	Meaning
0	Continue normal processing
4	Continue normal processing VMF has been modified
8	Reject the initialization of this tape

Parameter List

At entry, Register 1 points to a parameter list containing the following:

Offset	Size	Description
0	4	Address of the command verb.
4	4	Address of the DBRECORD.
8	4	Address of the current tape labels.
12	4	Address of the DBRECORD for the mount volume.
16	4	Address of the OWNER data.

Specifications

NONREENTRANT

AMODE 24

RMODE 24

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU013).

TLMSXOPN - OPEN Exit

The OPEN function of the CA TLMS operating system management module TLMSOSMM allows you to access its 408-byte transaction record through this user exit routine so that you can change or add data to the record. The data can be examined later by the online recorder user exit, TLMSXUPD.

This exit is activated by specifying YES or *exitname* in the OPNEXIT system option.

Use

TLMSXOPN is taken from module TLMSOPEN, the data set PROTECT/OPEN function of TLMSOSMM. The user exit as supplied performs no function, and requires the insertion of your own code.

Register Usage

R0

work register

R1

address of TRANREC

R2

address of 3-word list - TCB, SVRB and ASCB

R3

free

R4

free

R5

free

R6

free

R7

address of OPEN DCB

R8

address of UC8

R9

address of IOSWA

R10

free

R11

address of options table TLMSIPO

R12

base register of TLMSXOPN - must code using R12

R13

exit save and work area

R14

return address

R15

work register

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU014).

This exit is activated by specifying YES for the OPNEXIT system option.

TLMSXSEC - Security Exit

When security is activated in CA TLMS, this exit is called just prior to the call made to the operating system security component. The exit can be used to perform several functions; 1) it can request a bypass of the system security call or allow the call to continue, 2) call parameters can be viewed or altered before system security is requested to perform its check.

This exit is activated by specifying YES or *exitname* in the SECEXIT system option.

Register Usage

TLMSXSEC is called before any system security request is performed. Register 1 contains the address of the CA TLMS system security parameter list. Register 2 contains the address of a 3-word list that points to the TCB, SVRB and ASCB. The code returned in register 15 indicates to the security module the action to take. A return code greater than zero indicates to bypass the security call and issue this return code to the original caller. Valid user return codes are 0-8; however, only 0, 4 or 8 should be used. Return codes greater than 8 will produce an error message and are reserved for the system security manager, TLMSSECU, which calls this exit.

Parameter List

The security parameter list passed to this exit is mapped by macro TLMWSECP, and is addressed by SECUPARM in the supplied sample exit. This parameter list is complete and is ready to be parsed and passed to the system security component on return from this exit call. This exit can view all the values in this list, and some can be altered to provide flexibility with different system security packages. For example, the SECOPN system option (check data set access at OPEN) is set to YES, and the security call is at data set creation. The access type byte (SCPACC) will have a value of X'20', indicating create access. This may, in fact, not work for the data set rules that have been coded for the installed security system, and may have to be altered to X'08' (SCPAUPD) for update.

User Exit Parameter Fields

SCPGLOBL

Global security flags can be viewed or altered by the exit.

SCPGLOG

X'40' - suppress event logging

SCPGMSG

X'20' - process security messages

SCPGABN

X'10' - abend, security violation

SCPFUNC

Security call function request

SCPACC

Access requested (default access is always READ), function request 3 only.

SCPRESLN

Resource length, function request 3 only.

SCPUSRID

Address of requesting user ID.

SCPPSWD

Address of user password

SCPNPSWD

Address of new user password

SCPTRMID

Address of terminal ID. This may or may not be available at user exit call time. Test for zeros before using contents as an address.

SCPACEE

ACEE/ACMCB. This may or may not be available at user exit call time. Test for zeros before using contents as an address.

SCPRESOR

Address of resource name if function call is 3.

SCPCLASS

Address of resource class name if function call is 3.

SCPPROGM

Address of program name if function call is 3.

SCPVTRM

Address of virtual terminal name if function call is 1. This may or may not be available at user exit call. Check for valid address.

SCPVPSWD

Address of re-verify password if function call is 7.

SCPVOLSR

Address of volume serial number if function call is 3 and resource is DSN.

SCPCALID

Calling routine ID as one of the following: OPEN, CLSE, TLTP, CMND, BTCH (batch).

The above fields or addresses may be viewed, if present, and will depend on the type of security call in progress. Address pointers should be checked for zeros, in which case no value was supplied and there is no default. Although it is recommended that none of these values be altered, changes to some may be desirable (for example, resource class name). Contact CA TLMS Technical Support when considering these types of changes.

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU015).

This exit is activated by specifying YES for the SECEXIT system option.

TLMSXTRS - Tape Retention System Exit

TLMSTRS passes control to a user-written exit routine (TLMSXTRS) at two places; preprocess (**before** the retention process), and scratch process (**after** TRS has determined the volumes should be scratched).

TLMSTRS calls this exit using standard operating system linkage. Register 1 points to a parameter list which contains

- Preprocess (one entry)

  address of the VMFBASE or VMFMDS record
- Scratch process (two entries)
  - address of the TLMSTRS work record
  - address of the TLMSXTRS message CAT4550E (60-byte user message)

This exit is activated by specifying YES or *exitname* in the TRSEXIT system option.

Preprocess

TLMSTRS allows you to access the CA TLMS Volume Master File base volume records and multi-data set records before the retention process. This allows you to modify fields in the records to cause TRS to process a record differently, or to ignore it entirely. When EOF is reached on the Volume Master File, your exit receives control with the entry in the parameter list pointing to a record containing X'FF's (high values). This allows you to close user files and do other cleanup operations your exit may require.

To cause a record to be ignored by TLMSTRS, simply change the field BATYPE (or MDTYPE) to a blank character.

Scratch Process

TLMSTRS allows you to access the CA TLMS work record just after TRS has determined that the volumes should be scratched. The volumes may then be allowed to continue in scratch status on return to TLMSTRS.

To cause the volumes to bypass TRS scratch processing, place a nonblank character in byte 12 of the 60-byte user message. On return to TLMSTRS, the volumes will not be set to scratch status. The TLMS041 report prints the controlling data set information of the bypassed volumes and the 60-byte user message. If byte 12 is a blank character, TRS scratch processing continues and the volumes will be scratched.

Include the following statement in your TRS procedure (CATTRS):

```
//CAIVMFI  DD DSN=CAI.TLMS.VMF,DISP=SHR
```

Your exit must always zero the return code in register 15 before returning to TLMSTRS.

The assembler DSECT for the Volume Master File base record is named TLMSVMFB, and is TYPE=BASE. The multi-data set record is TYPE=MDS and the multivolume record, TYPE=MVL, maps to the same DSECT.

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU016).

This exit is activated by specifying YES for the TRSEXIT system option.

TLMSXUPD - Volume Master File Update Exit

TLMSXUPD lets you access the 408-byte CA TLMS transaction record to examine and change data before updating the Volume Master File. It also lets you control the volume's output use (data set protection).

The user exit must be an executable load module in the CA TLMS target library and must be named TLMSXUPD.

Specify YES or *exitname* in the UPDEXIT system option to activate this exit.

Use

The user exit will gain control for automatic functions and manual updates. The exit is passed the address of the update transaction record and the address of the Volume Master File base volume record. You will receive control in the following cases:

1. At CLOSE/EOV of an output tape data set.
2. At CLOSE/EOV of an input tape data set.
3. At OPEN of an output data set during the volume mount and verification phase.
4. For manual updating (TSO, TLTPISPF, INQR, and TRS).
5. At OPEN of an input data set.

The DSECT for the update transaction record is named TLMSTRAN.

TLMSUPDT calls your exit using standard operating system linkage. Register 1 points to a parameter list which contains the

- address of the 408-byte transaction record
- address of the volume master file base volume record

You must set the return code (in register 15) to zeros when returning from the exit. If set to a nonzero value, the transaction is ignored, and no update takes place.

T6SUBTYP must be set to either 1 (reject tape) or 0 (accept tape). If T6SUBTYP is other than one of these two values, the online recorder will abend with a user code of 997. T6NONVMF=# indicates that either EXPDT=98000 or SPACE=(1,(1,1)) was used to indicate a foreign volume, or the volume is not in the VMF. T6RESVOL=Y indicates the volser is in the VMF; T6RESVOL=N indicates that the volser is not in the VMF.

You can control the data set protection feature of CA TLMS by examining both the TRANREC record and the VMFBASE record. If you decide to reject the volume (disallow OPEN for output), place the character 1 in the field T6SUBTYP and set R15 to a nonzero value. TLMSUPDT will then return to the operating system OPEN module and force a dismount of the volume. If you decide to allow the volume to be opened for output, but do not want CA TLMS's data set protection logic to be executed, place the character 0 (zero) in the T6SUBTYP field and set R15 to a nonzero value. TLMSUPDT will then return to the operating system OPEN module and allow the volume to be used.

Note: When a foreign volume condition exists for cases 1, 2 or and 3, X'FF' (high values) is placed in the Volume Master File base volume record.

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU017).

CTSUXEDM - Defining External Data Manager Tapes

If the CA TLMS External Data Manager (EDM) interface is implemented, this exit expands the ability of the interface to define data sets which should be flagged by CA TLMS as externally managed. However, with the ability to assign multiple EDM IDs through the selection criteria in the CTOEDMxx member of CAI.CTAPOPTN with full pattern masking support, the need for this exit is minimized. Use of this exit may be justified if data set name, job name, program name and ddname masks are not enough to satisfy EDM selection.

CTSUXEDM is called from the SVC *after* one of the specified EDM masks has been found to match a tape request.

This exit is activated by specifying YES or *exitname* in the EDMEXIT system option.

Register Usage

CTSUXEDM uses standard linkages. Register 13 points to a save area, Register 14 is the return address, and Register 15 is the entry point address of CTSUXEDM. On return to the SVC module, Register 15 should contain one of the return codes below. Branching to one of the supplied labels will cause the appropriate return code to be set.

Label	Return Code	Meaning
RTNCDE00	00	EDM controlled, use the EDMID as assigned.
RTNCDE04	04	Do not use the selected EDMID, but continue the search of possible EDMIDs for another match.
RTNCDE08	08	The EDMID has been modified by the user exit. This modified EDMID should be used.
RTNCDE12	12	Not EDM controlled. Do not use the selected EDMID, and do not continue the search.

Parameter List

At entry, Register 1 points to a parameter list containing the following and is mapped by CTMEDMLK:

Fieldname	Size	Contents
EDDSN	4	Data set name
EDJOB	4	Jobname
EDDPGM	4	Program name
EDDD	4	DD name
EDSSNAME	4	"TMS" or "TLMS"
EDEDMID	8	New EDM ID
EDOPT	4	Options when no SSCT
	12	Reserved

Specifications

REENTRANT

AMODE 31

RMODE ANY

Note: If this exit is used, as with all realtime exits, it must be in CTAPLINK.

Example

A sample of CTSUXEDM is included with CA TLMS. Registers are saved, a new save area is GETMAINed and addressability is established.

CTSUX1G - Scratch Subpool Verification at Open

This exit is called during realtime processing (OPEN output) to verify the tape mounted in response to a nonspecific mount request is from the correct CA TLMS scratch tape subpool. It is also called when Realtime Stacking is selecting a volume to have files stacked on.

The CA TLMS WTO SSI may have assigned a scratch subpool name when the original mount message was issued. The WTO SSI uses rules defined in CTOSCRxx and CTONSMxx loaded by TLMSRIM and pattern masking specifications to assign a scratch subpool based on data set name, SMS management class, job name, and unit address. If the original CA TLMS mount message is not modified by WTO user exits, MPF processing, or other means, this exit is not required. If the original CA TLMS mount message is modified, then this exit can be used to verify that the tape mounted is from the correct subpool.

Four exit points are provided in CTSUX1G:

RETURN00

Continue scratch subpool verification, no changes made/volume is eligible for Realtime Stacking.

RETURN04

Continue scratch subpool verification, user exit made changes.

DISMOUNT

Demount tape with reject code supplied in field AXREJECT.

REJECT

The tape is not eligible for Realtime Stacking.

This exit is activated by specifying YES or *exitname* in the SPLEXIT system option.

Parameter List

The parameter list passed in Register 1 is mapped by the CTMUXPRM macro. The addresses and areas applicable to CTSUX1G for subpool verification are:

Fieldname	Contents
AXMPOOL	Scratch pool of mounted tape
AXAPOOL	Scratch pool assigned via NSM Rules
AXREJECT	Reject code to dismount volume

Note: AXAPOOL may be modified. On return from this exit, AXMPOOL (scratch pool of mounted tape) and AXAPOOL (scratch pool derived from NSM rules) are compared. If they are not equal, an NS80 or NS84 reject is issued.

The addresses and areas applicable to CTSUX1G for Realtime Stacking validation are:

Fieldname	Contents
AXAPOOL	Scratch pool assigned via NSM rules
AXADBREC	Address of common tape record of volume selected
AXREJECT	If non-zero, the volume will not be selected for Realtime Stacking

Note: If AXREJECT is non-zero on return, the specified volume will not be selected for Realtime Stacking and another volume will be selected. Exit CTSUX1G will again be called once another eligible volume is found.

Specifications

REENTRANT

AMODE 31

RMODE ANY

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (CTSJUX1G).

TLMSRACF - CA TLMS Scratch Exit

Use TLMSRACF to specify the actions to be taken when a tape volume is scratched. TLMSRACF provides the exit with the address of the volume record for the tape being scratched.

The user exit must be an executable load module in LINKLIST or a library available to the CTS address space.

TLMSRACF can be used to support RACF CLASS='TAPEVOL' and delete the volume profile. TLMSRACF can also be used to request tape robots to set the volume to scratch status. TLMSRACF *cannot* be used to prevent TLMS from scratching the tape volume.

TLMSRACF uses the following registers:

- R0 – R10 Work registers
- R11 Volume being scratched
- 12 Base register of TLMSRACF
- R13 Exit save and work area
- R14 Return address
- R15 Work register

This user exit should be assembled and linked through SMP USERMOD. An example of the JCL is located in CAI.CTAPJCL (TLMJU003).

To activate this exit, specify YES or exitname for SCREXIT system option.

More information

IBM 3494 Tape Library (see page 179)
IBM 3494 Virtual Tape Server (VTS) (see page 182)
IBM RACF Security Setup (see page 317)
RACF Interface (see page 326)

Chapter 5: Scratch Pooling

This section contains the following topics:

About Scratch Pooling

CA TLMS Scratch Pooling includes several features which enhance the tape processing facilities of the operating system. These include:

- The Scratch Pool Management feature allows you to restrict nonspecific output access to tape pools of CA TLMS controlled volumes.
- The Specific Mount Message option provides for message prefixing of specific output requests to indicate tape volume status in the VMF.
- CA TLMS may not allow the use of a tape volume for input or output for various reasons.
- NL and BLP processing requires additional processing including the request for the volume serial number.
- Nonresident tape volume processing may require additional information from the operator.
- Catalog Interface to keep the CA TLMS database updated with the current OS catalog status for each CA TLMS controlled tape file.
- The Real-time Stacking feature allows you to define scratch pools that support the stacking of multiple files on each volume in the pool without requiring application JCL changes.

CA TLMS console messages are detailed in the *Message Reference Guide*.

Using Scratch Pool Management

The CA TLMS Scratch Pool Management feature allows you to protect defined ranges of tapes in the VMF by restricting nonspecific output access based on:

- Data set name
- SMS Management Class name
- Creating job name
- Creating unit name
- Expiration date
- Temporary file status
- Specifying the pool name in the JCL

Many applications require the use of tapes with specific physical characteristics, such as tape length, quality of tape, reel size, and type of container. The requirements can be imposed by government agencies or internal business procedures or for operating convenience. The multiple scratch pool management capabilities of CA TLMS provide the necessary support to satisfy these requirements in a fully automated manner.

Describing the Scratch Pool and Access Rules

Scratch pools are assigned by adding control statements to two CAI.CTAPOPTN members:

- CTOSCRxx defines the tape pool name and the range(s) of volumes assigned to the tape pool
- CTONSMxx defines the rules for access to a tape pool

These two members are read by TLMSRIM during CA TLMS initialization.

Request a scratch pool by specifying the scratch pool name in the JCL VOL=SER= parameter, as shown:

```
//ddname   DD DSN=tape.dsname,DISP=(NEW,CATLG),UNIT=cart,
//         VOL=SER=poolnm,LABEL=(1,SL)
```

The scratch pool name must be six characters or less. If you use scratch pooling, you must specify at least one scratch pool assignment control statement in the CTONSMxx member. If the CTONSMxx member references at least one scratch pool, you do not need assignment control statements for scratch pools that are specified in the JCL.

Once CA TLMS is installed, you can maintain these members using your in-house text editing facilities.

Restrictions

CA TLMS assigns tape pools based only on the first file (primary data set) of the volume. Secondary data sets receive the same pool assignment as the primary data set for the same volume set. CA TLMS does not assign subpools for foreign tapes, which includes (EXPDT=98000 or SPACE=(1,(1,1))).

In the case of multivolume, multi-file data sets, the assignment for a new scratch volume is identical to the previous volume, which in turn can be traced back to the primary data set assignment on the first volume.

For multivolume files, when a new volume is opened during output, the pool assignment is the same as for the preceding volume. If the assignment rules have been changed since the creation of the original volume, the assignment of the new volume is based on the new assignment rules.

The following hierarchy of scratch pool assignment is observed:

1. &&TEMP and DSN are temporary
2. Specification of the scratch pool through the VOL=SER=pool JCL parameter
3. Assignment through an SMS Management Class
4. Assignment through External Data Manager rules
5. Assignment through the DSN, JOB, UNIT, RETPD and EXPDT parameters

Data sets which are not assigned to a tape pool by the assignment rules are automatically assigned to a generic tape pool which contains all tapes not defined to a tape pool.

Mount Messages

After activated by TLMSRIM, the CA TLMS WTO subsystem passes control to CA TLMS for all tape mount messages. If the mount request is nonspecific and requires a tape from a specific pool, the mount message is modified to include the tape pool identification (up to 13 characters). If the mount request is specific, the mount message reflects the location of the volume, and whether it is a CA TLMS controlled volume.

In all cases, a reissued operating system mount message is prefixed with a CA TLMS message identification to notify the operator that the intercept is present and functioning. The original and reissued WTO appears on the SYSLOG and JESLOG; only the new WTO is displayed on the console.

On a system console, a nonpool mount message is prefixed with CTS001, while a pool-controlled mount is prefixed with CTS002.

For example, a MINIREEL scratch pool is assigned the range, MI0001-MI9999, and data sets using a high-level alias of MINI are authorized to write to the MINIREEL pool. When the mount request is issued, CA TLMS modifies the mount message to read:

```
*CTS002 IEF233A M 05A0,MINIREEL,SL,6250,JOBNAME,STEP1,MINI.DATASET.NAME
```

For tape mounts not reflecting a data set name prefixed with MINI, the mount message reads:

```
*CTS001 IEF233A M 05A0,SCRTCH,SL,6250,JOBNAME,STEP1,NON.MINI.DATASET.NAME
```

JES3 messages use the prefix CTS004 for non-pool mount messages, and CTS005 for pool-controlled mounts.

Note: If using STK silos, you must have an entry coded above the STK silo entry in the subsystem name table. The entry is TLMS.

Scratch Pool Enforcement

When Scratch Pool Management is enabled, CA TLMS provides enforcement for both pool and nonpool nonspecific output requests. If a data set requires the use of a specific pool, but the volume mounted to satisfy the request is not part of the assigned range(s), CA TLMS demounts the volume and issues a NOT SCRATCH message. The request is reissued.

If the request is not for a specific pool, but a volume is mounted that is assigned to a specific pool, CA TLMS demounts the volume and issues the message. The request is reissued.

3480/3490 Cartridge Tape Drive Support

A sample exit allows the first eight characters of the tape pool name to be displayed on 3480/3490 cartridge tape drives. The exit disables the automatic cartridge loader (ACL). This exit prevents the ejection of all tapes in the ACL when a NOT SCRATCH condition occurs from Scratch Pool Management. For information about the installation of this exit, see the *Installation Guide.*

Specific Mount Message Processing

If the mount message is for a volume that is CA TLMS controlled and the volume status in the VMF indicates that it has been checked out of area, the following mount message is issued:

```
*CTS007  IEC501A M 0E82,X00007(L2;BA46),,JOBNAME,STEPNAME,DSNAME
```

The location and cabinet/slot are displayed within the message above.

If the mount message is for a volume that is nonresident (not defined in the VMF), the following mount message is issued:

```
*CTS008  IEC501A M 0E82,X00001,,JOBNAME,STEPNAME,DSNAME
```

If the requested volume is CA TLMS controlled, and the volume status in the VMF indicates that it is inhouse, the following message is issued:

```
*CTS009  IEC501A M 0E82,X00011,,JOBNAME,STEPNAME,DSNAME
```

Scratch Pool Management Definition

The Scratch Pool Management feature lets you define the ranges or a subrange in the VMF as a scratch tape pool

Restrict output access to the pool based on the SMS Management Class, data set name, job name, and/or unit name.

When CA TLMS intercepts a scratch-pool-controlled nonspecific mount request, it modifies the text of the mount message. The name reflects the pool name instead of the SCRTCH or PRIVAT that the system generates when mounting a volume from the correct pool.

If a scratch-pool-controlled request is generated and the mounted volume is not in the correct range, CA TLMS demounts the invalid volume and reissues the mount request. If a nonpool controlled request is generated and the tape mounted is assigned to a scratch pool, it is demounted and the request reissued.

The READONLYnnnnn subpool definition is also allowed. If the first eight characters of the subpool name are READONLY (followed by any five characters), the tapes in that subpool can be opened for READ processing only. No output is allowed.

Note: Never point a CTONSMxx rule to a READONLYnnnnn subpool.

Use this function when:

- You are testing a peer-to-peer (grid) configuration of a virtual tape system.
- You do not want to allow updates to any production tapes while in Disaster Recovery.

Note: Do not suppress the IBM mount messages that CA TLMS uses to process through the MPF exit. These message IDs are IAT5210, IAT5624, IEC101A, IEC501A, IEC501E, IEF233A, IEF233D, IEC704A, IEC534D, and IEC507D. If they are suppressed, CA TLMS honors the suppression bit and the messages are not handled. If the mount messages are being suppressed, acratch subpooling does not function properly.

You can enable each scratch pool for Real-time Stacking. Because file stacking is done at the pool level, ensure that all files directed to a stacking subpool have common characteristics, such as retention, vaulting, and security. You can dynamically stack any file that is eligible for normal direction to a specific scratch pool onto a volume already containing active files. with three restrictions:

1. The label type must be SL.
2. The file is not temporary.
3. The file sequence is 1 (default) and a specific volume was not coded in the VOL=SER= field.

A volume can be stacked if it has other files that are dynamically stacked with a higher file sequence number and with a stacking enabled scratch subpool. The volume must meet the following restrictions:

- Not marked as nonstackable
- Not closed by abend
- Not out-of-area
- Not an EDM-controlled volume
- Not part of a multivolume chain

If the next available volume is in use, it is bypassed and the next available volume is selected. TLMS does not delay a job waiting to be dynamically stacked.

Specification:

Control statements must be supplied in two members in CAI.CTAPOPTN.

- CTOSCRxx describes the pool names and the range of tapes assigned.
- CTONSMxx describes the SMS Management Class or combination of data set name, creating job name, and unit name that can access a pool that is defined in CTOSCRxx.

These two members are read during the CA Common Services TLMSRIM initialization of CA TLMS and their contents are verified before implementing the option.

Name Control Statement (CTOSCRxx)

The Name Control statement must be supplied in CTOSCRxx. The format is:

```
SCRPOOL=name,[STACK=YES/NO,]
[MAXFILES=nnnn,] [THRESHOLD=yyy,]
[SNEXT=SPECIFIC/SCRATCH,]
[NCAT=STACK/NEXT,]
RANGE=lowvol-highvol [,RANGE=lowvol-highvol]
```

SCRPOOL

Identifies the control statement and must be the first keyword.

name

The name that you assign to this scratch tape pool. The name can be up to 13 characters but must not contain a blank, comma, equal (=) sign, or single quote.

STACK=YES/*NO*

Indicates whether to activate file stacking for this pool.

MAXFILES=nnnn

Used if stacking is active for this pool. Specifies the maximum file sequence number that can be dynamically assigned at file open time. Values can be 1-9999. Default=200

THRESHOLD=nnn

Used if stacking is active for this pool. Specifies the volume use that, if reached during close processing, makes the volume no longer eligible for dynamic stacking. Values can be 1-100 and can reflect the percent of capacity that is used for the media type as identified in the VMF. Default=80

NCAT=*STACK*/NEXT

Used if stacking is active for this pool. Indicates what to do to files that do *not* have CATALOG specified as their normal disposition in the JCL. STACK indicates that these files are eligible like cataloged files. NEXT indicates that they are not eligible for file stacking (file sequence is 1). The SNEXT option is used.

SNEXT=*SPECIFIC*/SCRATCH

Used if stacking is active for this pool. Indicates what to do in all of the following cases:

1. NCAT=NEXT and the file does not have CATALOG in second disposition.
2. No eligible active volumes were found in pool to stack files on.

If SNEXT=SPECIFIC is specified, a specific VOLSER calls the next scratch tape in the pool. The scratch subpool is searched for the next volume in scratch status.

If SNEXT=SCRATCH is specified, a normal CTS002 message asks to mount a scratch tape from the specified pool.

RANGE=lowvol-highvol

Indicates the external volume serial number in the pool.

lovol

Lowest external volume serial number in the pool range

hivol

Highest external volume serial number in the pool range

The RANGE keywords must be the last keywords for the pool.

If either MAXFILES or THRESHOLD is reached, the volume is marked as "no longer eligible for stacking".

To continue a scratch pool definition statement, end the statement with a complete parameter followed by a comma, and continue on to the next statement beginning in position 1 or later. The number of continuations that are allowed has no limit, but only 255 ranges per tape pool are permitted.

You can name Scratch pools with any combination of up to 13 EBCDIC characters. Do not use a blank, comma, equal sign, or single quote. Because it appears in modified mount messages, make the name meaningful to the operator, as illustrated by the following examples:

```
SCRPOOL=100000-109999      Any tape with a volume serial number in
       the
                     range of 100000 through 109999
SCRPOOL=BLUE.STRAP  Tape with a blue strap
SCRPOOL=1200.FOOT   Any 1200 foot tape reel
SCRPOOL=POOL.A             Any tape from tape pool A
SCRPOOL=CERTIFIED   Specially certified tape
SCRPOOL=BACKUP.TAPE Special backup-only tape pool
SCRPOOL=SVC.DUMP    Pool for SVC dump tapes only
```

The choice of names is unlimited and can reflect your operating environment.

The members of a single tape pool do not need to be contiguous volumes in the tape library. Each tape pool can contain a number of ranges of volume serial numbers. However, a VOLSER cannot be specified in more than one subpool definition. For each range required, specify:

```
RANGE=lowvol-highvol
```

The values for lowvol and highvol can specify the same volume serial number, resulting in a single-volume tape pool.

Note: If STACK=YES and MAXFILES=1, then every scratch request within the pool follows the SNEXT keyword. If you use SNEXT=SPECIFIC, all scratch requests are turned into specific volser requests but no dynamic file stacking is performed.

Name Control Examples:

Example 1

```
SCRPOOL=SCRPOOL.A,RANGE=W10000-W10099,RANGE=Y09900-Y09999,
    RANGE=000000-009999
```

Example 2

```
SCRPOOL=000000-009999,RANGE=000000-009999
```

Example 3

```
SCRPOOL=BLUE,RANGE=X99000-X99099,RANGE=X89000-X89049
```

Example 4

```
SCRPOOL=STACKM,
    STACK=YES,THRESHOLD=70,
    RANGE=201000-202000
```

Example 5

```
SCRPOOL=100000-100099,RANGE=750003-750187
```

Examples 1 through 4 are correct examples, reflecting both the external nature of the scratch pool and its contents. Example 5 shows the type of error to avoid. The external description of the scratch pool implies a different set of volumes than the RANGE parameter defines.

To direct an output request to a specific scratch subpool using JCL, the pool name must be six characters or less. See Example 3. Code the pool name as if it were the VOLSER.

Note:

- The unit that is specified must be able to satisfy at least one of the tapes from the specified pool.
- You can specify only one pool name in the VOL=SER= field. You cannot add other VOLSERs except for the scratch subpool name.
- You cannot have two or more DD statements in the same step pointing to the same scratch subpool through the VOL=SER= specification. You can have two or more DD statements using VOL=SER=poolid, but they must be different pool names.

```
//ddname    DD DSN=tape,dsname,DISP=(NEW,CATLG),UNIT=cart,
//          VOL=SER=BLUE,LABEL=(1,SL)
```

Assignment Control Statement (CTONSMxx)

To define scratch assignment rules, generate control statements specifying the poolname and either SMS Management Class or a combination of data set name, creating job name, and unit name. The format for the assignment control statement is:

```
DSN=dsname,JOB=jobname,POOL=poolname,UNIT=unitname
```

Or

```
MGMTCLAS=classname,POOL=poolname
```

DSN

This assignment applies to only those data sets whose data set name is the specified dsname, unless further restricted by the JOB or UNIT parameter. If DSN is coded, MGMTCLAS cannot be specified. Code DSN=&&TEMP to assign temporary data sets or work tapes to scratch pools. The data set name can be fully qualified or you can use the CA TLMS Pattern Masking facility to specify it. For more information, see Pattern Masking (see page 161).

JOB

This assignment applies only to data sets being created with the specified job name. If JOB is coded, MGMTCLAS cannot be coded. The job name can be qualified or specified using the CA TLMS Pattern Masking facility.

UNIT

This assignment specifies the unit physical name (ccuu). Esoteric names are not supported. If UNIT is coded, MGMTCLAS cannot be coded. The unit name can be qualified or specified using the CA TLMS Pattern Masking facility. Enter Three character units with a leading zero.

MGMTCLAS

This assignment specifies the 1-8 character SMS Management Class name. The SMS Management Class name cannot be coded using CA TLMS pattern masking specifications. If MGMTCLAS is coded, DSN, JOB, and UNIT cannot be coded.

POOL

Assignments that are made under this use tapes from the tape pool specified by poolname.

To continue a scratch pool assignment statement, end the statement with a comma before position 72 and continue on to the next statement, beginning in position 1 or later. The number of continuations that are allowed has no limit.

Important! If you change the production CTONSMxx member and REINIT CA TLMS to make them active, add the new entries to the very end of the current member for the real-time tape processing not to be affected.

Assignment Control Statement Examples:

To assign a data set to a tape pool every time the data set is created, enter:

```
DSN=dsname,POOL=poolname
```

Make this assignment regardless of the creating job or unit.

If local naming conventions call for backup data sets to have names of NIGHTLY.BACKUP.volser.generation, the nightly backups can be assigned to a specific pool by:

```
DSN=NIGHTLY.BACKUP.-,POOL=backup.pool
```

To use a special pool for IMS or CICS online log tapes:

```
DSN=ims.log.dsname,JOB=ims.jobname,POOL=ims.log.pool
DSN=cics.log.dsname,JOB=cics.jobname,POOL=cics.log.pool
```

If naming conventions require that all test jobs begin with a character, such as a T, you can restrict programmer test tapes to a specific range by entering:

```
DSN=-,JOB=T-,POOL=test.poolname
```

Assignment is made by determining whether the combination of dsname and job name matches any of the entries read during the initialization. For multiple matches, such as:

```
DSN=ABCD-,POOL=C,UNIT=0E8?
DSN=ABC-,POOL=D,UNIT=0E??
```

A data set named ABCDE is assigned to pool C, because ABCD- is a more qualified match than ABC-. For two equal matches on a data set name, use the entry with the more exact job name or unit name or both.

External Data Manager Specifications

If an External Data Manager (EDM), such as DFHSM, is to control the CA TLMS tapes, you must supply the information about the tapes.

An externally managed tape is as follows:

- Created through an authorized program (APF authorized), and the program is linked AC=1.
- Matches the selection criteria for one of the External Data Manager IDs (EDMID) specified in the CTOEDMxx member of CAI.CTAPOPTN.

An EDMID is an eight-character field that is stored in the VMF Volume record to identify which EDM controls that tape. Within this member, you can specify multiple EDMIDs. Duplicate EDMIDs are allowed. The same EDMID can identify multiple sets of definitions.

Specifications

The following parameters are specified in the CAI.CTAPOPTN member, CTOEDMxx. If more than one option (except EDM) is specified, an *AND* condition is assumed (all specifications must be met). A mask can be provided for PGM, DSN, JOB, and DD.

DD

The ddname (up to eight characters)

DSN

The data set name (up to 44 characters)

EDM

A name identifying the EDM (up to eight characters)

JOB

The job name (up to eight characters)

PGM

The program name (up to eight characters). Must be APF authorized.

Multiple EDMs can be specified in this member. Full pattern masking is supported.

The following example illustrates how all DFHSM managed tapes would be placed under EDM control:

```
EDM=HSM1,DSN=*.MIGTAPE.-,PGM=ARCCTL
EDM=HSM1,DSN=*.DUMP.-,PGM=ARCCTL
EDM=HSM1,DSN=*.BACKTAPE.-,PGM=ARCCTL
```

The Scratch Pool Management feature allows a scratch pool to be assigned to an EDM. The scratch pool assignment control statement that is defined in member CTONSMxx of CAI.CTAPOPTN supports the specification of an EDM ID with the scratch pool name:

```
EDM=HSM1 ,POOL= HSM1VOLUMES
```

EDM=HSM1 is assigned the scratch pool HSM1VOLUMES. When a tape belongs to the EDM HSM1, output scratch processing requests volumes from the HSM1VOLUMES scratch pool. This scratch pool must be defined to the CTOSCRxx member. See Using Scratch Pool Management (see page 150) for further scratch pool management.

Default

Member CTOEDMxx is distributed with null values. If no EDM is used, no action is required. You can use the user exit CTSUXEDM to define data sets managed externally.

Chapter 6: Pattern Masking

This section contains the following topics:

About Pattern Masking

Pattern Masking can be used for scratch subpool definitions. The CA TLMS Pattern Masking feature provides a method to identify and select data sets based on a mask, or *pattern.* A pattern is a character string consisting of a combination of alphanumeric characters into which may be incorporated special characters that perform unique masking operations during match and compare functions. Patterns can be sorted in a most to least-specific order to ensure the accuracy of a match.

CA TLMS Pattern Masking Feature

Once patterns are created, the CA TLMS utility CTSPMTST can be used to test the patterns by executing COMPARE, MATCH and VALIDATE operations. CTSPMTST processing produces printed output, which aids in resolving pattern errors.

The following terms are used in this discussion:

Nodal

A pattern mask or object that contains node separators, such as data set name.

Nodeless

A pattern mask or object that does not contain node separators, such as job name.

Object

A data element that the pattern is to be matched against. This is typically a data set name or job name.

Wild card

Any one (or more) of the special masking characters used in a pattern to represent the presence or absence of the character(s).

Pattern Characters

The following special characters can be used as wild cards in the definition of a pattern:

Question mark (?) X'6F'

This character is used to match any character, or the absence of a character in the object. It cannot match a node character.

Exclamation mark (!) X'5A'

This character is used to match any character, but not the absence of a character in the object. It cannot match a node character.

Pound or hash sign (#) X'7B'

This character is used to match a **numeric** value in the object. It cannot match the absence of a character.

At sign (@) X'7C'

This character is used to match an **alphabetic** character in the object. It cannot match the absence of a character.

Asterisk (*) X'5C'

This character is used to match any number of characters (including zero) within a node. It cannot match the node character.

Dash (-) X'60'

This character is used to represent any number of characters (including zero) before and after its position in the pattern. It can match the node character.

Back slash (\) X'E0'

This character is used to specify that the character immediately following it is to be taken literally; that is, * would indicate that the asterisk is to be matched as an asterisk, and not used as a wild card character. A back slash is not valid at the end of a pattern unless it is preceded by a back slash (\\).

Node character

This character is used to define a node separator. Typically, this character is a period, as used in a data set name.

Pattern Hierarchy

To make the most accurate match when multiple patterns are involved, patterns must be arranged in a hierarchy from the most- to the least-specific. The following classifications are used to assign the accuracy of a pattern. These classifications apply to objects that contain nodes only. (For information on the classification of nodeless objects, see the topic, Nodal versus Nodeless Objects.

Specific

This classification defines any pattern that does not contain a dash. Patterns so classified are sorted left to right. Examples are:

```
ABC.MNO.XYZ
ABC.X*.LIST
ABC.??LIST??.*DATA
```

Prefixed/Suffixed/Containing Combination

This classification defines any pattern containing two or more dashes within the pattern, but not at the beginning or end of the pattern. An example is:

```
ABC-LMNO##-XYZ
```

Prefixed/Suffixed Combination

This classification defines any pattern that contains a single dash (or * for nodeless) within the pattern, but not at the beginning or end. Examples are:

```
ABC???.-XYZ
ABC-XYZ
```

Prefixed

This classification defines any pattern that ends with a dash. Examples are:

```
ABC-
ABC-.XYZ.-
*ABC.-
A!B??.-
```

Note: It is appropriate to note that some of these examples represent inefficient patterns (those that require more scanning overhead), as is shown in the two examples above. Although patterns that contain embedded dashes or begin with a wild card character are valid, it is recommended that you attempt to be as specific from left to right as possible to avoid this overhead. This note also applies to the next two pattern classifications, *Suffixed* and *Containing.*

Suffixed

This classification defines any pattern that begins with a dash. Examples are:

```
-.LIST*
-.*XYZ
-.???XYZ
```

Containing

This classification defines any pattern that begins and ends with a dash. A pattern containing an additional embedded dash also falls under this classification. Examples are:

```
-XYZ-
-.??ABC??.-
-.-.-
```

Once classified, patterns in the same class can be compared directly. The pattern is translated to force wildcard characters to be higher than other characters in the pattern mask. Wildcard characters that match are processed in the following collating sequence:

- Pound sign (#), most specific
- At sign (@)
- Exclamation mark (!)
- Question mark (?)
- Asterisk (*)
- Dash (-), least specific

The following list illustrates this order:

```
ABCDEF.XYZ
ABC###.XYZ
ABC@@@.XYZ
ABC!!!.XYZ
ABC???.XYZ
ABC*.XYZ
ABC-.XYZ
```

Nodal versus Nodeless Objects

Nodal objects differ from *nodeless* objects primarily in the use of node separators and use of the asterisk. The asterisk is equivalent to the dash in a nodeless object. Since there are no node boundaries, the only boundary respected by the asterisk is the end of the object. In a nodal object, the asterisk terminates at the end of a node.

All other aspects of pattern masking are the same.

Object Classes

Predefined pattern classes have been established to define the characteristics of the objects to be matched. These classes define whether the object is nodal or nodeless, what characters to recognize as node separators, and the size of the object. The following classes are predefined for the CA TLMS pattern masking facility:

MVSFILE

Indicates that the object is a z/OS file (data set) name. The object is defined as nodal, 44 characters in length, and uses the period (.) as a node separator.

MVSJOB

Represents any z/OS name that has the same characteristics as the job name. The object is defined as nodeless and eight bytes in length. Other objects in this class include DDnames, step names, PROC names, program names, and so forth.

Pattern Matching Routine

The pattern matching routine performs three basic functions:

- Pattern validation
- Pattern matching (pattern to object)
- Pattern hierarchy (pattern to pattern)

Pattern Validation

Pattern validation verifies the contents of a pattern. This function primarily checks for invalid combinations. The following combinations are invalid because they are ambiguous:

```
?*    ?-    *?    **    *-    -?    -*    --
```

A back slash (\) cannot be used as a wildcard character at the end of a pattern. It is valid at the end only if preceded by a back slash (\), which indicates that it is to be taken literally as a back slash.

Pattern Matching

The pattern supplied is matched against the object, the match being based on the wildcard characters found in the pattern. This routine is called recursively to scan for the current *segment* from the pattern. A segment is one of the following:

ABC.

Portion up to node separator (nodal)

ABC?

Portion up to (excluding) question mark

ABC*

Portion up to (excluding) asterisk

ABC-

Portion up to (excluding) dash

ABC

Portion up to end of pattern

The following examples illustrate segments:

```
SSDDEV.\@#@!?.A*C.*.-XYZ
SSDDEV.  \@#@!   ?.  A   *C.  *.  -XYZ
```

Recursion is necessary to ensure that an accurate match is made when using the wildcard characters ?, * or - to prefix a pattern, such as:

```
???ABC
*ABC
-ABC
```

Consider the following pattern and object (nodal):

```
Pattern:     ABC-XYZ.DATA

Object:      ABC.LMNOXYZ.TUV.XYZ.DATA
```

The pattern matches an object beginning with ABC and ending with XYZ.DATA. When the pattern is matched, the first segment to be matched is ABC. The matching routine looks for ABC at the current location. Since it matches, it moves to the next segment, -XYZ. The dash indicates that the object is to be scanned for the string, XYZ. This is found in the object within the second node, LMNOXYZ.

The next segment to be compared is DATA. Since the next node of the object is TUV., a mismatch occurs. The scan routine must step back to the original segment that started the scan, -XYZ., and continue. This scan would stop at the fourth node of the object, XYZ.. Since the next segment of the pattern and the object would both match, the pattern would be considered to match the object.

If the last node of the object had been something other than DATA, the comparison would have failed.

Scanning Using the Question Mark (?), Asterisk (*) and Dash (-)

Scanning is performed for specific text (including the at sign (@), pound sign (#) and exclamation mark (!)) following a question mark, an asterisk or a dash. The question mark represents the maximum number of characters that can be mismatched before the scan is terminated.

The scan is terminated when the pattern is located, the end of the node is encountered, or the maximum number of mismatches has occurred. The asterisk is used to scan for the text that follows it. The scan ends when the text is located or the end of the node is encountered. The dash is similar to the asterisk, with the exception that the scan after a dash continues until the test is found or the end of the object is encountered.

The following example illustrates a scan using the question mark:

```
Object:    AAABC.XYZ
Pattern:   ????BC.XYZ
Object segment:    AAABC.
Pattern segment:   ????BC.
```

This pattern indicates that as many as four characters can be mismatched at the beginning of the pattern before a mismatch is declared on the entire pattern.

Pattern Hierarchy

This function provides the capability to arrange patterns in a most specific order. It furnishes the mechanism for comparing patterns, but does not actually sort them. The application is responsible for placing them in order based on the results of the compare. Pattern Hierarchy is detailed earlier in this discussion.

Pattern Masking Examples

The following examples are supplied to assist in understanding how patterns are used.

TWJRK.CA?.LOAD	matches	TWJRK.CA.LOAD TWJRK.CAS.LOAD
	but not	TWJRK.CA11.LOAD
TWJRK.??CA?.LOAD	matches	TWJRK.CA.LOAD TWJRK.CAS.LOAD TWJRK.TCAS.LOAD TWJRK.T2CA.LOAD TWJRK.T2CAS.LOAD
	but not	TWJRK.CA11.LOAD TWJRK.T22CAS.LOAD
TWJRK.T?MS.LOAD	matches	TWJRK.TMS.LOAD TWJRK.TLMS.LOAD
	but not	TWJRK.XTMS.LOAD TWJRK.TMSX.LOAD
TWJRK.CA!.LOAD	matches	TWJRK.CYS.LOAD
	but not	TWJRK.CA.LOAD TWJRK.CA11.LOAD
TWJRK.!!CA!.LOAD	matches	TWJRK.T2CAS.LOAD

	but not	TWJRK.CA.LOAD TWJRK.CAS.LOAD TWJRK.TCAS.LOAD TWJRK.T2CA.LOAD TWJRK.CA11.LOAD TWJRK.T22CAS.LOAD
TWJRK.CA!?.LOAD	matches	TWJRK.CAS.LOAD TWJRK.CA11.LOAD
	but not	TWJRK.CA.LOAD
TWJRK.?!CA?.LOAD	matches	TWJRK.TCAS.LOAD TWJRK.T2CA.LOAD TWJRK.T2CAS.LOAD
	but not	WJRK.CA.LOAD TWJRK.CA11.LOAD TWJRK.CAS.LOAD TWJRK.T22CAS.LOAD
TWJRK.CA##.LOAD	matches	TWJRK.CA11.LOAD
	but not	TWJRK.CAS.LOAD TWJRK.CASA.LOAD TWJRK.CA150.LOAD
TWJRK.@@@@.LOAD	matches	TWJRK.TLMS.LOAD
	but not	TWJRK.CAS.LOAD TWJRK.TMS.LOAD TWJRK.CA11.LOAD TWJRK.CATMS.LOAD
TWJRK.\@\@.LOAD	matches	TWJRK.@@.LOAD
	but not	TWJRK.CA.LOAD
TWJRK.*.LOAD	matches	TWJRK.CA.LOAD TWJRK.CAS.LOAD TWJRK.CA11.LOAD TWJRK.TLMS.LOAD TWJRK.CA11R500.LOAD
	but not	TWJRK.CAS.R50.LOAD TWJRK.LOAD

TWJRK.CA*.LOAD	matches	TWJRK.CAS.LOAD TWJRK.CA11.LOAD TWJRK.CA11R500.LOAD
	but not	TWJRK.CAS.R50.LOAD
TWJRK.*CA*.LOAD	matches	TWJRK.CA.LOAD TWJRK.CAS.LOAD TWJRK.TCAS.LOAD TWJRK.T2CA.LOAD TWJRK.T2CAS.LOAD TWJRK.CA11R500.LOAD TWJRK.TLMSIICA.LOAD
	but not	TWJRK.X.CA.LOAD TWJRK.CA.X.LOAD
TWJRK.A*B*C.LOAD	matches	TWJRK.ABC.LOAD TWJRK.ABXC.LOAD TWJRK.AXBC.LOAD TWJRK.AXBXC.LOAD TWJRK.ABXXXXXC.LOAD TWJRK.AXBXXXXC.LOAD TWJRK.AXXBXXXC.LOAD TWJRK.AXXXBXXC.LOAD TWJRK.AXXXXBXC.LOAD TWJRK.AXXXXXBC.LOAD
	but not	TWJRK.ABCX.LOAD
TWJRK.-.LOAD	matches	TWJRK.CAS.LOAD TWJRK.CAS.R50.LOAD
	but not	TWJRK.LOAD
TWJRK-LOAD	matches	TWJRK.LOAD TWJRK.TESTLOAD TWJRK.CAS.LOAD TWJRK.CAS.R50.LOAD
ABC-XYZ	matches	ABCXYZ ABC12XYZ ABC12.TUVXYZ
	but not	ABC12.XYZZ

TWJRK.\@#A!?.A*C.*.-XYZ	matches	TWJRK.@1AZ.ABXXC.LMNOP.ABXYZ.ABCXYZ TWJRK.@9A1Z.AXXXXXXC.LMNOP.ABC1XYZ
	but not	TWJRK.11AZ.ABC.LMNOP.XYZ TWJRK.@1AZ.ABC.XYZ TWJRK.@1AZ.ABC.XYZ.XYZA

More Information:

Pattern Characters (see page 162)

Defining SMS Management Classes for Tape Data Sets

This section shows how you can use CA TLMS to complement System Managed Storage (SMS) in a z/OS environment. You will see how to modify SMS Automatic Class Selection (ACS) routines to define a SMS management class for TAPE files with or without a Storage Group of TAPE defined. By assigning an SMS Management Class to a tape file, CA TLMS can determine scratch subpools, assign file retention, and enforce SMS Management Class retention limits.

Note: The only requirement is that SMS must be active and the CA TLMS option, SMS, is set to YES in TLMSIPO. This implementation of SMS does **not** require that DFSMS 1.1 or higher be installed.

Requests to the SMS Subsystem Interface are sent by CA TLMS to obtain the definition of a Management Class and to invoke the Data Class, Storage Class and Management Class routines. A Data Class and Storage Class must be assigned to invoke the SMS Management Class ACS routine.

Once a Management Class is returned, it is stored in the VMF. The Management Class is saved in the field xxSMSMGT in the Data Set record. The Data Class and Storage Class names are not saved. The Management Class can be manually changed using UPD vvvvvv MGMTCLAS=.

SMS ACS Services are invoked:

- At OPEN OUTPUT time, with &ACSENVIR. set to CATAPEO
- At Mount Message time, with &ACSENVIR. set to CATAPEP
- At Scratch Pool Validation time, with &ACSENVIR. set to CATAPEP

The SMS data may be used by TRS User Exit to set retention.

How to Code ACS Rules to Affect Tape Processing

Assign a Storage Class

```
/*************************************************************/
/* ASSIGN A STORCLAS TO APPL01 TAPE DATA SETS                */
/*************************************************************/

    WHEN (&HLQ='APPL01'.  &.& &ACSENVIR='CATAPEO')
       SET &STORCLAS. ='BASE'
    WHEN (&HLQ='APPL01'.  &.& &ACSENVIR='CATAPEP')
       SET &STORCLAS. ='BASE'
    WHEN (&HLQ='APPL01'.  &.& &ACSENVIR='CATAPE')
       SET &STORCLAS. ='BASE'
```

Assign a Management Class

```
/*************************************************************/
/* ASSIGN PAYROLL GDG DATASETS ON TAPE TO CATALOG            */
/*************************************************************/

 WHEN (&HLQ. = 'PAYROLL' &.& &DSTYPE = 'GDG'
  && &ACSENVIR = 'CATAPEO') SET &MGMTCLAS. = 'CATALOG'

/*************************************************************/
/* ASSIGN PRODUCTION TAPE DATASETS TO THE PRODUCTION POOL (PROD)*/
/*************************************************************/

  WHEN (&HLQ. ¬= 'TEST' &.& &ACSENVIR = 'CATAPEP')
    SET &MGMTCLAS. = 'PROD'
```

Chapter 7: TLMSRIM - Initializing and Reinitializing CA TLMS

This section contains the following topics:

About TLMSRIM

TLMSRIM uses the facilities from CAIRIM to initialize and reinitialize CA TLMS. It is responsible for installing the CA TLMS SVC, building the TLMS subsystem, initializing the WTO subsystem interface, loading realtime modules to common storage if not found in the link pack area, installing the CA TLMS OSI to track tape activity, locating and anchoring CA TLMS realtime user exits, building an optional realtime retention table, and installing the SMF interface to capture tape read and write errors.

TLMSRIM is a required phase of the CA TLMS initialization process. At system startup time, the CAIRIM procedure invokes the TLMSRIM module when the control statements are used as input to the PARMLIB DD of CAIRIM. CAIRIM invokes TLMSRIM via an alias name for the release. The alias has the form of TLrrINIT, where the rr is a hex release and modification.

Important! This program should be used only by systems programmers responsible for maintaining CA TLMS.

Note: If you have other vendor products which modify the operating system's data management modules, and these modified versions are MLPA, you must include the CAG8LIB2 DD in the CAIRIM procedure. Any MLPA data set must be specified in the CAG8LIB2 DD which contains these modifications. If you have a data set name for the contents of the LPA other than SYS1.LPALIB, you must put that data set name in the CAG8LIB1 DD and include the DD in the CAIRIM procedure. Ensure that the CAIRIM procedure (CAS9) has security access (READ) to the libraries containing the operating system's data management modules.

Note: If you want TLMSRIM to build the optional realtime retention table, you must include the CAIRMF DD in the CAIRIM procedure. The CAIRMF DD points to the active "retention master file" (RMF).

Job Control Statements

The following control statement is specified in the PARMLIB DD of the CAIRIM procedure when TLMSRIM services are required:

```
PRODUCT (CA TLMS/-PRD)
    VERSION(TLC6)
    INIT(TLC6INIT)
    PARM(INIT [,OSI=NO] [,LPA=xxxxxxxx] [,SMF=YES])
    LOADLIB(authorized.library)
```

This control statement directs TLC6INIT which is an alias of TLMSRIM to initialize CA TLMS. If the module is not found in the link pack area, TLMSRIM will load it in extended common storage, using CAIRIM dynamic SVC services, and update the SVC table. The INIT function can be performed only once. Subsequent invocations of TLMSRIM are required to specify the REINIT parameter. (See the following syntax).

The OSI=NO keyword specifies that the CA TLMS OSI, used to track realtime tape activity, is not to be loaded. If the CA TLMS OSI is not loaded, no tape jobs should be run since tape activity will not be intercepted. This parameter should not be used unless you are certain tape activity will not occur. For more information on the CA TLMS OSI, see Control Statements.

Once the initial TLMSRIM has run, the REINIT parameter can be used to reinitialize certain components of CA TLMS.

The following control statement is specified in the PARMLIB DD of the CAIRIM procedure:

```
PRODUCT (CA TLMS/-PRD)
    VERSION(TLC6)
    INIT(TLC6INIT)
    PARM(REINIT
        [,LPA=xxxxxxxx]
        [,WTO=CLEAR|FORCE]
        [,OSI=YES|NO|REINIT]
        [,SMF=YES])
    LOADLIB(authorized.library)
```

The REINIT parameter will reinitialize the components specified by the other keywords provided in the CAIRIM PARM control statement.

Control Statements

INIT

Directs TLMSRIM to perform CA TLMS Initialization. Can be used only once.

LOADLIB

Specifies that the LOADLIB can be loaded from a specific authorized load library instead of from the link list concatenation.

LPA=

The LPA= statement directs TLC6INIT to load the specified resident module or modules into common storage.

OSI=

Specifies that the CA TLMS OSI is to be loaded, not loaded, or refreshed.

NO

Specifies that the CA TLMS OSI should not be loaded.

YES

Specifies that the CA TLMS OSI will be loaded.

REINIT

Specifies that the CA TLMS OSI should be refreshed. All tape processing MUST be stopped while the OSI is being reinitialized.

REINIT

The REINIT keyword is used on subsequent TLMSRIM requests to reinitialize or refresh certain components of CA TLMS.

ALL

Specifies that all CA TLMS commonly accessed, resident routines are to be reloaded, and CA TLMS pointers refreshed.

modname

Specifies the name of the module to be added to common storage.

SMF=YES

This specifies that TLMSRIM should attempt to load and install any SMF exit modules.

WTO=

This control statement directs TLMSRIM to either temporarily or permanently shut down the CA TLMS WTO and DOM subsystem interface (WTO SSI). The WTO SSI is used to capture and reissue tape mount messages if CA TLMS tape scratch pools are implemented.

CLEAR

Temporarily disables the WTO SSI. A subsequent TLMSRIM REINIT request will re-enable the WTO SSI.

FORCE

Shuts down the WTO SSI for the remainder of the life of the IPL. The WTO SSI will not be reactivated until the system is IPLed again.

Control Statement Examples

For the initial execution of TLMSRIM during system startup time, use the following control statements.

- To load a new version of the CA TLMS real-time user, exit CTSUXEDM:

  ```
  PRODUCT(CA TLMS/-PRD) VERSION(TLC6) INIT(TLC6INIT)
  PARM(REINIT,LPA=ALL,OSI=REINIT)
  LOADLIB(authorized.library)
  ```

  Specify the LOADLIB parameter if the new version of CA TLMS is not found in the current link list concatenation. The REINIT refreshes the stored address of the TLMS modules.

- To shut down temporarily the WTO SSI *only:*

  ```
  PRODUCT(CA TLMS/-PRD) VERSION(TLC6) INIT(TLC6INIT)
  PARM(WTO=CLEAR)
  ```

- To reactivate the WTO SSI:

  ```
  PRODUCT(CA TLMS/-PRD) VERSION(TLC6) INIT(TLC6INIT)
  PARM(REINIT)
  ```

  The REINIT refreshes pointers in the CA TLMS resident table. Resident routines and user exits are relocated and the SMF exits are reinitialized.

Chapter 8: Automated Tape Libraries and Virtual Tape Support

This section contains the following topics:

About Tape Processing

Mainframe tape processing has evolved over the last five decades from the prevalence of labor-intensive manual tape processing to Automated Tape Libraries (ATL) and Virtual Tape systems (VTS). Virtual tape systems have further increased efficiency by eliminating the need to write to physical tapes at all, instead writing tape data to "virtual" volumes in a disk cache. At a later time these virtual volumes are consolidated and stacked on physical tape volumes and exported from the virtual tape system for disaster recovery or other purposes. CA TLMS has supported each new phase in tape technology as it has been introduced and will continue to support the latest tape media types, robotics, and virtual tape systems.

Automated Tape Libraries and Virtual Tape systems are now the standard in most installations. Customers have adopted ATLs for the following benefits:

- Faster mount times
- Minimize tape handling to reduce the possibility of operator error
- Easier tracking of tape inventory

The introduction of virtual tape extended these benefits by also providing:

- Improved utilization of physical tape cartridges
- Instantaneous dismount and rewind processing

These benefits have driven the acceptance of robotics and virtual tape systems as the standard for efficient tape processing today.

This section is organized by tape library vendor for both virtual tape and basic ATLs. Note that the terms ATL, library, and robot will be used interchangeably in this section. No difference between the terms is intended.

VMF Fields Used to Track Tapes in Automated Tape Libraries

CA TLMS provides two fields in the Volume Master File (VMF) to track the residency of tapes in ATLs and virtual tape systems and a third field used by virtual tape solutions to record the physical volume serial number that a virtual tape is written to when exported.

- ATLTYP-Automatic Tape Library type (also known as the robot type)
- ATLNUM-Automatic Tape Library number (also known as the robot number)
- ACTUAL_VOLSER-Actual volume serial number (also known as the container volume)

ATLTYP

The Automatic Tape library type identifies the manufacturer (IBM, STK, and others) and identifies if the ATL is a virtual tape system. Specific bit settings are defined for the leading robotics and virtual tape vendors. A partial list follows; the complete list is defined in Appendix D in the *User Guide* for TAPEDB field ROBOT_TYPE. This field is zero if the volume is not in an ATL.

- X'01' IBM
- X'02' STK
- X'81' IBM Virtual Tape System
- X'82' STK Virtual Storage Manager
- X'88' CA Vtape

ATLNUM

Automatic Tape Library device number - is used to distinguish between multiple robots of the same type in the installation. This field is assigned by the customer. Typically, if an installation has three robots of the same type they are assigned the identifiers 1, 2, and 3. They could however be assigned any number up to 254.

ACTUAL_VOLSER

Actual volume serial number, also described as the *container* volume serial number - is used to record the physical volser that one or more virtual volumes are written on. This field is not used for basic ATL support and is updated only when the volumes are exported from the virtual tape system.

The ATLNUM and ATLTYP fields can be displayed using the DVA command and updated with the UPV command. The ACTUAL_VOLSER field is updated internally within robotic interface processing only. The ACTUAL_VOLSER field cannot be updated by the user. These fields are also displayed using CA Earl reports and the CA Vantage GMI interface, however the key updaters of these fields are the user exits provided by CA or the robotic system vendor. These exits are invoked when volumes are entered into or removed from the library to provide up to the minute status on volume residency. CA TLMS does not update these fields. They are updated only if the entry and eject exits are installed.

The CA TLMS Tape Retention System (TRS) alters its processing for both real and virtual tapes that are in ATLs. As TRS processes the retention schedule for a tape, it does not prevent movement because the tape is in an ATL. Instead, TRS creates a "Move Report" entry which shows the tape moving from its location with the "cabinet/slot" as follows:

- **ATL**-indicates a physical tape in an ATL
- **VTS**-indicates a virtual tape in an ATL
- **VTX**-indicates a virtual tape that has been exported from an ATL

For a physical tape the TO location has a cabinet/slot assigned for the tape. VTS and VTX tapes have no cabinet/slot assigned and are shown as VTS or VTX. This continues through the retention schedule.

IBM 3494 Tape Library

The IBM TotalStorage 3494 Tape Library is IBM's main cartridge tape library for the mainframe. The IBM 3494 also supports connectivity to open systems and is the framework for IBM's virtual tape system, the IBM TotalStorage 3494 Virtual Tape Server (VTS). This section describes how to configure CA TLMS with the IBM 3494 Tape Library. For information on configuring CA TLMS with the IBM 3494 VTS, see the section entitled "IBM 3494 Virtual Tape Server (VTS)" below.

Integration between CA TLMS and the IBM 3494 Tape Library (ATL) or the IBM 3494 VTS is based on four exits that are used to communicate with the robot:

- CBRUXENT-IBM OAM Volume Entry
- CBRUXEJC-IBM OAM Volume Eject
- CBRUXVNL-IBM OAM Volume Not In Library
- TLMSRACF-CA TLMS Scratch Exit

Object Access Method (OAM) is a part of IBM's DFSMSdfp component and is designed to provide management of data objects rather than files. OAM is also used to manage IBM's tape libraries as part of an overall storage hierarchy, based on DFSMS System Managed Storage concepts and facilities. OAM includes an address space that runs on z/OS which provides object related services for applications as well as entry, eject, and related processing for tape libraries. The OAM address space also maintains the Tape Configuration Database (TCDB) which defines the IBM tape volumes and libraries managed. The TCDB is made up of one or more VOLCATs, a special type of ICF.CATALOG which is specifically marked as a volume catalog.

OAM includes a component, Library Control Services (LCS) which is used to manage the movement and tracking of tapes in IBM automated tape libraries. CA TLMS interfaces to OAM through requests made to LCS.

To configure your installation to access the IBM 3494 for basic type library support you must customize the OAM exits CBRUXENT and CBRUXEJC. The CBRUXVNL exit is optional. CA recommends that the CA TLMS scratch exit TLMSRACF be installed to synchronize scratch status between CA TLMS and TCDB, however this may also be achieved by implementing batch synchronization procedures. A discussion of the processing in each exit follows. Note that additional calls to these exits are performed when an IBM 3494 VTS is being managed.

The additional calls related to virtual tape processing are discussed in the section entitled "IBM 3494 Virtual Tape Server (VTS)" below.

IBM 3494 CBRUXENT Volume Entry Exit

The OAM CBRUXENT exit is executed whenever a tape cartridge is inserted into the ATL. The CBRUXENT exit provided by CA TLMS will determine the status of the tape within the VMF and then allow or disallow the entry of the tape, and set the status in the TCDB to scratch or private, based on the status in the VMF. When installed, the CBRUXENT exit provided with CA TLMS will update VMF fields ATLTYP and ATLNUM to indicate the tape is in the ATL. By default a foreign tape cannot be entered into the ATL. If this is attempted, the tape will be left in "insert" status. This exit must be customized to specify the SMS name of each robot and to assign an ATL number in an internal table.

IBM 3494 CBRUXEJC Volume Eject Exit

For basic 3494 ATL processing the OAM CBRUXEJC exit is invoked when a tape is physically ejected from the ATL. The sample CBRUXEJC exit provided by CA TLMS will clear the ATLTYP and ATLNUM fields to indicate that the volume does not reside in any robot. For basic 3494 ATL support, no changes in the CBRUXEJC sample exit are required.

IBM 3494 CBRUXVNL Volume Not in Library

The OAM CBRUXVNL exit is invoked when a requested tape is not found in the ATL, providing you an opportunity to insert the tape into the ATL and retry allocation without cancelling the job. Use of this exit is strictly optional. If you choose to use this exit you should be aware that OAM also calls this exit for DASD that is offline and for all tapes (even round reels) not in an SMS managed IBM robot.

No updates to CA TLMS or OAM are performed based on this exit. The only time you would need to install and use this exit is if you wanted to allow operations the chance to insert a cartridge back into the 3494 and re-drive the allocation instead of simply allowing the job to have a JCL error or abend. If you have a 3494 ATL with similar devices both inside and outside the robot (so any tape can be mounted either inside or outside the 3494), then there is no need to use this exit.

This exit must be customized for your environment. You must determine what processing is to be done if the tape you are requesting is not in the library. For example, if you have 3490 drives outside the 3494 and have only 3590 drives inside the 3494, you would want to un-comment out two lines that compare the CA TLMS returned density with the value "3590." If you do not un-comment out these two line, then ALL allocations for the 3494 would be driven through the CBRUXVNL exit.

CA TLMS TLMSRACF Scratch Exit

The TLMSRACF scratch exit for CA TLMS is called every time a volume is put into scratch status. This exit may also be customized to keep the OAM TCDB synchronized with the VMF when volumes are scratched. Simple customization of this exit is required to activate the call to LCS to synchronize scratch status. Member TLMJU003 of CAI.CTAPJCL is provided for this purpose. It contains sample JCL to apply USERMOD TLMU003.

The OAM TCDB may also be synchronized by executing the CTSSYNC utility, described below. Use of the TLMSRACF interface provides immediate synchronization of volumes as they are scratched and reduces or eliminates the need to run the CTSSYNC utility for synchronization purposes (it will still need to be run for eject processing).

CA TLMS CTSSYNC Utility Support for the IBM 3494 ATL

The CTSSYNC utility is used to communicate with OAM through LCS services for both basic 3494 ATL and the 3494 VTS. It is the main interface module between CA TLMS and OAM. For basic 3494 ATL support, CTSSYNC is used to eject tapes and can also be used to synchronize the TCDB with the VMF. Additional functionality to support the 3494 VTS is described in the section entitled "IBM 3494 Virtual Tape Server (VTS)" below.

CTSSYNC may perform eject or synchronous functions on either a range of volsers or a list of volsers. CA TLMS provides CA Earl report member TLERPT21 of CAI.CTAPEARL to read the VMF after TLMSTRS has run and format eject statements for volumes selected for movement to an off-site location. These eject statements are then processed by CTSSYNC.

IBM 3494 ATL Recommendations and Procedures

Install the OAM user exits CBRUXENT, CBRUXEJC, and CTSUXVNL using sample job CTSJUCBX from CAI.CTAPJCL. CTSJUCBX is a USERMOD to apply the exits to the IBM SMP/e zone.

Usermod TLMU003 may also be run to provide immediate scratch synchronization with the OAM TCDB. The SMP/E statements to apply TLMU003 are provided in member TLMJU003 of CAI.CTAPJCL.

You must also add two new jobs (or job steps) to your normal daily CA TLMS utilities to create and process an eject list to select volumes for off-site movement. CA Earl report, TLERPT21, is provided in CAI.CTAPEARL to create the list of volumes to eject. Add a new job to execute TLERPT21 after TLMSTRS has been run. A second job should be added to execute CTSSYNC to process the eject list created by TLERPT21. As cartridges are returned from off-site locations you can insert them back into the 3494 ATL.

Note that if you run the IBM 3494 ATL in a Peer-to-Peer configuration (with a second IBM 3494 ATL offsite mirroring your 3494 on-site) you will not need to run the TLERPT21 or CTSSYNC EJECT jobs to create an eject list. In Peer-to-Peer configurations if you have activated the TLMU003 usermod to provide immediate scratch synchronization, the elapsed run time of TLMSTRS may be extended due to delays in communication with the off-site ATL. If the elapsed time for TLMSTRS becomes too long, you may choose to remove the TLMU003 usermod and run CTSSYNC with a parameter of SYNC to cause the status of recently scratched volsers to be synchronized with the VMF. This will result in the appropriate scratch volumes being updated in the OAM TCDB to scratch status.

IBM 3494 Virtual Tape Server (VTS)

This section documents how to configure CA TLMS to work with the IBM TotalStorage 3494 Virtual Tape Server (VTS). For information on the basic IBM 3494 ATL and a general introduction to CA TLMS robotic support, see the section entitled "IBM 3494 Tape Library" above.

As with the IBM 3494 ATL, integration between CA TLMS and the IBM 3494 VTS is based on four exits that are used to communicate with the robot:

- CBRUXENT-IBM OAM Volume Entry
- CBRUXEJC-IBM OAM Volume Eject
- CBRUXVNL-IBM OAM Volume Not In Library
- TLMSRACF-CA TLMS Scratch Exit

IBM VTS - CBRUXENT Volume Entry Exit

The OAM CBRUXENT exit is executed whenever a tape cartridge is inserted into the ATL or when a virtual volume is imported from a physical tape. The CBRUXENT exit provided by CA TLMS will determine the status of the tape within the VMF and then allow or disallow the entry of the tape, and set the status in the TCDB to scratch or private, based on the status in the VMF. When installed, the CBRUXENT exit provided with CA TLMS will update VMF fields ATLTYP and ATLNUM to indicate the tape is in the ATL. By default a foreign tape cannot be entered into the ATL. If this is attempted, the tape will be left in "insert" status. This exit must be customized to specify the SMS name of each robot in an internal table for both 3494 ATL and 3494 VTS implementation.

IBM 3494 VTS - CBRUXEJC Volume Eject EXIT

The CBRUXEJC exit is invoked when a physical tape is ejected and when a virtual tape is exported. This exit will need to be modified if you are using the 3494 VTS with the import and export functions. A pseudo DSN will need to be defined so that CA TLMS can track the physical tapes used during the export, otherwise a default name will be generated. If you are using a 3494 VTS without the import/export feature, or if you are using a 3494 ATL, then no modification is needed to the CA TLMS supplied version.

During an export function, the following actions are taken on the virtual volumes being exported

- The ATLNUM field is set to zero
- The BOX/CCSS field is set to 'VTX'
- ACTUAL_VOLSER will be updated with the physical volser that the virtual volumes are being exported to
- The ASCTYP (Associated Type) field is set to 'V'

The following actions are taken on the physical volume (also known as the container volume)

- The ATLTYP field is set to VIBM
- The ATLNUM field is set to zero
- The ACTUAL_VOLSER and FSTVOL (First volume in chain) fields are set to the physical volser
- The ASCTYP (Associated Type) field is set to 'V'

- The DSN is set to the pseudo DSN defined in CBRUXEJC. If no DSN is defined in CBRUXEJC the following pseudo DSN will be used-"CATAPE.HOLDS.EXPORTED.VIRTUAL.TAPES.Vxxxxxx" where xxxxxx is replaced with the physical volser
- The KEEP DATE and EXPIRATION DATE is set to PERM
- The CREATING JOB is set to "OAM" and the CREATING PROGRAM is set to "TLMSQSTS"
- The VOLUME OWNER is set to 'VIBM'
- The LOCATION and RETURN LOCATION are set to the Data Center (DC)
- CREATION DATE and TIME are set to the current date and time

The pseudo DSN assigned to the physical volume should be added to the Retention Master File (RMF) to define your desired retention schedule for container volumes that consist of exported virtual volumes.

IBM 3494 VTS - CBRUXVNL Volume Not in Library

The OAM CBRUXVNL exit is invoked when a requested tape is not found in the ATL, providing you an opportunity to insert the tape into the ATL and retry allocation without cancelling the job. Use of this exit is strictly optional. If you choose to use this exit you should be aware that OAM also calls this exit for DASD that is offline and for all tapes (even round reels) not in an SMS managed IBM robot.

No updates to CA TLMS or OAM are performed based on this exit. The only time you would need to install and use this exit is if you wanted to allow operations the chance to insert a cartridge (or import a virtual volume) back into the 3494 and re-drive the allocation instead of simply allowing the job to have a JCL error or abend. If you have a 3494 VTS without the IMPORT/EXPORT feature there is no need to use this exit (since all virtual volumes will always be inside the robot).

This exit must be customized for your environment. You must determine what processing is to be done if the tape you are requesting is not in the library. For example, if you have 3490 drives outside the 3494 and have only 3590 drives inside the 3494, you would want to un-comment out two lines that compare the CA TLMS returned density with the value "3590." If you did not un-comment out these two lines, then ALL allocations for the 3494 would be driven through the CBRUXVNL exit.

CA TLMS TLMSRACF Scratch Exit Support for the IBM 3494 VTS

The TLMSRACF security exit provided by CA TLMS is appropriate for both 3494 ATL and 3494 VTS environments. It is not required, however it may be installed to keep the OAM TCDB synchronized when a volume is put into scratch status by CA TLMS. Simple customization of this exit is required to activate the call to LCS to synchronize scratch status. Member TLMU003 of CAI.CTAPJCL is provided for this purpose.

The OAM TCDB may also be synchronized by executing the CTSSYNC utility, described below. Use of the TLMSRACF interface provides immediate synchronization of volumes as they are scratched and reduces or eliminates the need to run the CTSSYNC utility for synchronization purposes (it will still need to be run for eject processing).

CA TLMS CTSSYNC Utility Support for the IBM 3494 VTS

The CTSSYNC utility is used to communicate with OAM through LCS services for both the basic 3494 ATL and the 3494 VTS. It is the main interface module between CA TLMS and OAM. CTSSYNC is used to eject tapes or to perform import/export functions for a VTS. It can also be used to synchronize the TCDB with the VMF. The CTSSYNC utility is documented in this guide.

When using CTSSYNC to import or export tapes from the VTS two additional DD statements are required-EXPORT DD and IMPORT DD. These DD statements must point to a unit in the VTS. This is where the volsers that are to be imported or exported are retained for passing on to LCS. Only one IMPORT or EXPORT can be run at a time.

CTSSYNC may perform eject, export, import, or synchronize functions on either a range of volsers or a list of volsers. CA TLMS provides two CA Earl report members in CAI.CTAPEARL to create control statements that can be processed by CTSSYNC:

- **TLERPT21**-Creates export or eject control statements by inspecting the VMF after TLMSTRS has been run.
- **TLERPT22**-This report can be used to create import control statements for a container volume specified as input. The control statements are processed by CTSSYNC to import the virtual volumes back into the VTS.

In addition, a CA Earl report is provided to create CA TLMS control statements to use in maintaining the VTS container volumes:

- **TLERPT23**-This report lists the exported virtual volumes and the container volumes they reside on. Tapes are listed by creation date within container volume. The scratch status of each tape is shown. This report may be used to determine when the virtual and container tapes should be re-imported into the ATL. Generally, they should be imported when all the virtual volumes are scratched. The container volume is created as an EDM and will not be automatically scratched. This report can generate the 'UPV volser SCRATCH=YES' command for each container tape whose virtual volumes are in scratch status or have been previously imported. Once the container volume has been scratched you will need to manually insert the physical tape into the library as an eligible stacking volume. Information on how to do this can be found in the IBM manual, *IBM 3494 VTS User Guide*.

For more information about CA Earl reports, see the *User Guide*.

IBM 3494 VTS without IMPORT/EXPORT - Recommendations and Procedures

Managing an IBM 3494 VTS without the IMPORT/EXPORT feature is similar to managing a basic IBM 3494 ATL; however since the volumes remain in the VTS, there is no need to define special rules for handling the physical container volumes. The OAM user exits CBRUXENT and CBRUXEJC should be installed using the instructions provided in the *IBM Tape Library Support* Manual. Source is provided in CAI.CTAPSAMP members CTSUXENT and CTSUXEJC that must be customized. Source is also provided for the optional Volume Not in Library exit in member CTSUXVNL.

Usermod TLMU003 may also be run to provide immediate scratch synchronization with the OAM TCDB. The SMP/E statements to apply TLMU003 are provided in member TLMJU003 of CAI.CTAPJCL.

Run your normal daily CA TLMS maintenance jobs. If you have implemented TLMU003 to provide immediate scratch synchronization with the 3494 VTS, when TLMSTRS runs it will notify the 3494 VTS which tapes have been scratched. All tapes (virtual volumes) remain within the 3494 VTS. If you do not implement TLMU003, you must add a job to your daily maintenance cycle to run CTSSYNC with a parm of SYNC after TLMSTRS runs. This will keep the OAM TCDB and the VMF scratch status synchronized.

IBM 3494 VTS with IMPORT/EXPORT - Recommendations and Procedures

Install the OAM user exits CBRUXENT and CBRUXEJC using the instructions provided in the *IBM Tape Library Support Manual*. Source is provided in CAI.CTAPSAMP members CTSUXENT and CTSUXEJC that must be customized. Source is also provided for the optional Volume Not in Library exit in member CTSUXVNL.

Usermod TLMU003 may also be run to provide immediate scratch synchronization with the OAM TCDB. The SMP/E statements to apply TLMU003 are provided in member TLMJU003 of CAI.CTAPJCL.

As discussed above, CA TLMS logically allows retention and movement to be performed for virtual tapes but does not assign cabinet/slots to them. They are scratched when their retention schedule is exhausted. As TRS processes the retention schedule for a tape, it does NOT prevent movement because the tape is in an ATL or VTS. Instead, TRS creates a 'Move Report' entry which shows the tape moving from its location. The 'cabinet/slot' field is updated as follows:

- ATL indicates a physical tape in an ATL
- VTS indicates a virtual tape in an ATL
- VTX indicates a virtual tape that has been exported from an ATL

For a physical (container) tape the TO location has a cabinet/slot assigned.

If you are vaulting data sets that are on virtual tapes and they go to different vaults, you will need to make some changes to CBRUXEJC and TLERPT21. In CBRUXEJC you will need to add logic to compare the DSN (and optionally other fields) and set different pseudo DSNs for the physical tapes to be created. TLERPT21 allows you to add the tape location to the EXPORT statement:

```
EXPORT,vvvvvv,L1
EXPORT,vvvvvv,L2
```

The vvvvvv is replaced with the VOLSER to be exported and L1 and L2 are examples of separate vaults you want physical volumes exported to. These names can be the vault name or any other meaningful destination for your site.

When CTSSYNC is run with these control statements, all tapes with L1 will be exported together, and all tapes with L2 will be exported onto another tape. This gives you the separation so that you can have exported tapes going to different vaults. If you are going to keep the physical tape offsite until it expires you do not need to add a vault pattern for the pseudo DSNs. If you need to keep them a number of days, then you need to have a new vault pattern with the pseudo DSN.

Procedures for Virtual and Physical Volume Maintenance when Using the IMPORT/EXPORT Feature

1. Run TLERPT23 to review the inventory of exported virtual tapes and the container volumes they reside on. TLERPT23 will also build "UPV volser SCRATCH=YES" control statements for TLMSINQR to expire physical tapes when all virtual volumes stacked upon them have been imported or scratched.
2. Run TLMSINQR to process with the control statements created by TLERPT23 *only* for those physical volumes with all virtual volumes in scratch status.
3. Run your daily CA TLMS TLMSTRS job to manage virtual volumes along with normal tapes.
4. Run TLERPT21 to identify the virtual volumes to be exported. These volumes have been identified by TLMSTRS because they have been scheduled for movement and their location is not in an ATL. TLERPT21 will build the CTSSYNC control statements to eject the physical tapes in an ATL and export the virtual tapes.
5. Run CTSSYNC to do the ejects and/or exports.
6. Run TLERPT22 to build the CTSSYNC control statements to do the IMPORT for the individual container volumes returned from an off-site location.
7. Run CTSSYNC using the control statements from TLERPT22.

IBM 3584 Tape Library

The IBM TotalStorage 3584 Tape Library is fully supported by CA TLMS using the same OAM exits and procedures as the 3494 Tape Library. The 3584Tape Library supports both open systems and zSeries attachments with support for the IBM Virtual Tape Server (VTS) and Peer to Peer configurations. The recommendations for sharing an IBM 3494 ATL or VTS also apply to the 3584when configured for use as a normal tape library or VTS.

For more information about configuring CA TLMS to work with the 3584 Tape Library, use the procedures and recommendations outlined in the IBM 3494 Tape Library and IBM 3494 Virtual Tape Server (VTS) sections.

Sharing an IBM 3494 ATL or VTS

The IBM 3494 ATL and 3494 VTS can be attached to multiple systems in two main configurations. The first is to "partition" the box (make it look like two separate boxes, each section attached to a different system) or you can share the 3494 ATL or 3494 VTS with different systems. In fact, you can partition the box and have one partition shared between some systems and have the second partition shared between different systems.

Normally, when the 3494 ATL or 3494 VTS is going to be shared between different operating systems; the box is partitioned. So one partition is used by z/OS (most likely shared by a couple of z/OS systems) and one partition is used by the other operating system (z/VM for example). In a partition mode, the 3494 operates as if it were two separate boxes and tape volumes cannot be shared. So tapes created in one partition cannot be used in the other partition.

For more information on sharing or partitioning a 3494 ATL or 3494 VTS, it is strongly recommended that you review the *DFSMS Object Access Method* Planning, *Installation, and Storage Administration Guide for Tape Libraries* manual (or "PISA" manual for short).

When sharing a 3494 ATL or 3494 VTS between z/OS systems there are four different ways to configure your CA TLMS and OAM environment:

1. Common CA TLMS VMF and common OAM TCDB (Tape Configuration Database, also known as the VOLCAT)
2. Separate CA TLMS VMF's and separate OAM TCDBs
3. Common CA TLMS VMF and separate OAM TCDBs
4. Separate CA TLMS VMF's and a common OAM TCDB

Each of these presents slightly different requirements and may meet different needs. Below are some of the items within the CA TLMS supplied exits that may need to be customized depending on your environment.

Common CA TLMS VMF and Common OAM TCDB

This is probably the most common way to share a 3494 ATL or 3494 VTS. All z/OS systems share a common VMF and a common TCDB. In this environment, very few modifications to the CA TLMS supplied entry and eject exits would need to be done. If you wanted to have the ability to enter foreign tapes into the robot (cartridges not defined to CA TLMS), you would have to change the entry (CBRUXENT) exit. In the CA TLMS supplied exit (CTSUXENT in the CAI.CTAPSAMP library) there is a small branch table after label "ASKCA;" where a branch to labels RC0, RC4, RC8, RC12, RC16, and RC20 is made. If you want to read foreign tapes inserted into the 3494, the easiest modification is to change the branch table so that in the case of a foreign tape the branch is made to RC8 instead of to RC16. So simply change "B RC16" to "B RC8" for foreign tapes. If you will not be reading foreign tapes inside the 3494 (for example, if the 3494 VTS has no native-attached physical devices) then there is no need to modify either the entry or eject exits supplied with CA TLMS other than to define your ATL(s).

Separate CA TLMS VMFs and Separate OAM TCDBs

This is probably the second most common way to share a 3494 ATL or 3494 VTS when the 3494 is not actually partitioned. It is almost the same as having the 3494 partitioned, but gives the user more flexibility in terms of adjusting the size (number of volumes and/or number of devices) on each "side" without having to redefine the partitions. For basic operation, the CA TLMS supplied entry and eject exits will not need to be modified at all. However, if foreign tapes are going to be inserted into the 3494 and used on one of the systems; then a more complicated version of the entry exit will be needed.

For an example, we will assume that CPUA and CPUB share a 3494 with separate VMFs and TCDBs, and we want to allow foreign tapes to be used as input on CPUA. In this case, the entry and eject exits for CPUB would be unchanged. Since CPUA cartridges would be foreign to CPUB, when CPUB is notified that a CPUA cartridge has been inserted it would simply ignore it. Then, when CPUA was notified it would assign the correct status (SCRATCH or PRIVATE) and take ownership of the cartridge.

However, in the same example the entry exit on CPUA would have to be modified and it would be more complicated. In the CA TLMS supplied entry exit (CTSUXENT in the CAI.CTAPSAMP library), there would have to be code added at label RC16. If the volume being inserted is defined to CPUB (which means a table would need to be created or a set of compares would need to be done to see if the volser matches those defined to CPUB), then the instruction "LA R15,UXEIGNOR" would need to be performed (followed by the branch to EXIT). However, if the tape is not defined to CPUB (again, based on the volser) then you would want to branch to label RC8 (which would put the foreign tape into PRIVATE status and take ownership of it on CPUA). Depending on the volser standards on CPUB, this could be very simple (look for a cartridge starting with "B0" or "B1") or very complicated.

Common CA TLMS VMF and Separate OAM TCDBs

This is not a recommended method to use, however it can be done. This gives the user the ability to logically partition the 3494 (with separate TCDBs) but have a common CA TLMS VMF. The problem is that the entry exit (CTSUXENT in CAI.CTAPSAMP) would need to be modified on both systems AND there would need to be additional modification to the TLMU003 usermod (the SCRATCH exit TLMSRACF). The entry exit on CPUA would need to have all the CPUB cartridges defined to it, and not take ownership of them (even though they are defined to the VMF). This additional code would best be done PRIOR to the "ASKCA" label, since you know that all the cartridges are known to both systems (since they share a common VMF). So, on CPUA before the ASKCA label you would have to compare the volser and see if it is owned by CPUB. If so, then branch to label RC16 (leave it in insert status for CPUB to take ownership). Likewise on CPUB, the exit would have to look for CPUA cartridges and branch to label RC16 for them (so that CPUA could take ownership of its cartridges). If there is a desire to share a common exit on both systems, then the exit will need to be modified to look at the SMFID or SYSTEM-NAME and make the determination if the volser should be owned or not based on a combination of which system it is running on and which tape is being inserted.

In this environment it would be best to not use the TLMSRACF scratch exit and to instead use multiple CTSSYNC jobs; one to process all scratch tapes owned by CPUA and the second to process all scratch tapes owned by CPUB. A CA Earl report would have to be written to select scratch tapes based on creating system SMFID (CPUA and CPUB) writing the output to separate transaction files. Then, on each system CTSSYNC would be run taking in the correct list of scratch tapes (CTSSYNC can be run with PARM=SYNC and the SYSIN would simply be a list of cartridges).

Separate CA TLMS VMFs and Common OAM TCDB

This is not a recommended method to use and should be avoided. Since CPUA and CPUB have a common OAM TCDB, then any tape could be mounted to satisfy a scratch request on either system. By having different categories defined on each system (this is done by modifying SYS1.PARMLIB members as explained in the OAM PISA manual), the problem of shared scratch pool could be removed. However, it would still be possible to request a specific volume from either system for any volume. Since each CA TLMS would only know about its tapes and not about the other system's tapes, they would appear as FOREIGN. So using EXPDT=98000 you could read (or update) active tapes from the other system. So in this case, you would have to tightly restrict the use of EXPDT=98000 processing. If not, then anyone that has access to EXPDT=98000 processing could request tapes from the other system and read and/or modify the existing data. Since full 44-character dataset name checking would NOT be available, external security rules would not be sufficient.

The entry and eject exits could be used as shipped with CA TLMS without modification. The TLMU003 usermod to implement immediate scratch processing could be implemented on both systems.

CA Vtape

CA Vtape uses your existing tape and DASD hardware to implement virtual tape with mainframe-class reliability and performance.

The CA Vtape interface to CA TLMS is internal and does not require user exits to implement. A CA Vtape parameter that is defined in the VTPARMS member of PARMLIB activates this internal interface. When the **TapeManagementSystem** parameter is set to TLMS or AUTO and CA TLMS is discovered on the system, CA Vtape activates the internal interface to provide the following benefits:

- The retention schedule for the virtual volumes comes from CA TLMS for use in backstore processing to group together virtual volumes with similar expiration attributes.
- During the externalization processing the container field (ACTVOL) in the virtual volume is updated in the VMF to reflect the physical volume it is written to.

In most cases, CA Vtape virtual volumes are handled as if they are real volumes. CA TLMS remains in control of scratch processing, which must be coordinated with CA Vtape. CA Vtape provides a job in CAI.INSTALL(TLMS) to perform scratch synchronization. Schedule this job to run after the daily TLMSTRS run.

The interface between CA Vtape and CA TLMS is documented in the *CA Vtape User Guide*. Complete planning and implementation steps are provided, but some of the main points include:

- Define an exclusive range of virtual volumes in the VMF for CA Vtape use.
- CA Vtape writes virtual volumes to the physical container volume during externalization processing as secondary files. CA TLMS writes the Multi-Data Set records (MDS) to contain information about secondary files. Ensure that you have extended the VMF to define enough MDS records to contain all the secondary file information that describes the virtual volumes.
- The MDS records are freed when all virtual volumes on the physical tape are expired.
- You can run the CA Vtape RECYCLE utility to recover fragmented physical tape space by copying active virtual volumes to new backstore tapes. This process also frees up MDS records.
- CA TLMS can report on the physical volumes that are externalized and the virtual volume to physical volume mapping. You can execute CA TLMS CAI.CTAPEARL member TLERPT23 to create this report. This information is important in disaster recovery planning.

StorageTek Tape Libraries

CA TLMS supports all major StorageTek (STK) tape libraries for mainframe environments. This includes but is not limited to the following list:

- PowderHorn
- TimberWolf 9740
- SL8500

CA TLMS integrates to STK libraries through the STK Host Software Component (HSC) Exit 6 (SLSUX06). HSC Exit 6 is documented in the *StorageTek HSC* System *Programmer's Guide*.

A sample SLSUX06 exit is provided in CAI.CTAPSAMP for customers who do not have an existing EXIT 6 routine and only need support for updating the ROBTY and ROBID. This version of the exit provides the basic code to update the ROBTY and ROBID when volumes are entered into the robot (define processing) and to clear ROBTY and ROBID when the volumes are ejected (delete processing).

StorageTek Virtual Storage Manager (VSM)

The StorageTek Virtual Storage Manager is integrated to CA TLMS through exits provided by STK to synchronize the VSM catalog with the VMF. This integration is documented in the STK documentation set.

Chapter 9: Utilities and Procedures

This section contains the following topics:

Overview

CA TLMS provides standard utility programs and procedures to perform initialization, backup, restore, and recovery for the Volume Master File, the VMF index file, the Retention Master File, the Alternate Log (ALOG) file, and the multiple access message queue file. (The VMF, VMF index, and RMF must be initialized when CA TLMS is installed; the other files only have to be initialized if you are using the optional features that require those files.)

Sample procedures for each utility detailed in this chapter can be found in the target (operating) procedure library CAI.CTAPPROC. They are listed below and discussed in this chapter in alphabetical order.

CATALOGB

Back up the Alternate Log file

CATALOGI

Initialize the Alternate Log file

CATCSMF

Create Volume Master File from SMF data

CATRMFE

Update for a full pattern masking RMF

CATRMFI

Initialize the Retention Master File (VSAM)

CATRMFP

Print program for full pattern masking RMF

CATVMFB

Back up the Volume Master File

CATVMFI

Initialize the Volume Master File

CATVMFID

Generate Volume Master File initialization parameters

CATVMFMR

Merge Volume Master Files

CATVMFRE

Reorganize the Volume Master File

CATVMFRL

Reload the Volume Master File

CATVMFRS

Restore the Volume Master File

CATVMFRV

Recover the Volume Master File

CATVMFSC

Set control record information in Volume Master File

CATVMFXI

Initialize or re-create the Volume Master File index

CTSDEU

Data Erase Utility

CTSTAPER

Test the tape management system

CTSPMTST

Validating Pattern Definitions

CTSSYNC

3495/3499 Tape Library Synchronization

TLMSNITT

Initialize tape volumes

Discussions in this chapter include a listing of the procedure JCL and the starter JCL, and a description of any output that may be generated. The examples use the system defaults, as provided on the CA TLMS installation tape.

CATALOGB - Backup the Alternate Log File

If the alternate log file is being used to log backup transactions, periodic backup of this file, using the CATALOGB procedure should be scheduled. Since tape processing need not be quiesced to perform this procedure, it is recommended that the alternate log file be backed up on a daily basis.

Process

Prepare the ++DUMP command as input to the CATALOGB procedure. Starting in position 1, the format of the command is

```
++DUMP
```

CATALOGB Procedure JCL

The following procedure is used to back up the alternate log file.

```
//*************************************************************
//*             ****  PROCNAME=CATALOGB  ****                 *
//*************************************************************
//** PROCEDURE TO DUMP THE CA TLMS ALTERNATE LOG FILE TO BACKUP *
//*************************************************************
//*
//CATALOGB PROC A='*',
//             ALOG='CAI.TLMS.ALOG',
//             BKUPALOG='CAI.TLMS.BKUPALOG',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             TAPE='TAPE'
//*
//BKUPALOG EXEC PGM=TLMSALOG
//*
//STEPLIB  DD  DSN=&LOAD.,DISP=SHR
//*
//TLMSOPTS DD  DSN=&OPTS.,DISP=SHR
//*
//CAIALOG  DD  DSN=&ALOG.,DISP=SHR
//*
//CAIBALOG DD  DSN=&BKUPALOG.,
//             DISP=(NEW,CATLG,DELETE),
//             UNIT=&TAPE.
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A.
//*
//SYSUDUMP DD  SYSOUT=&A.
//*
//SYSIN    DD  DUMMY
//*
//*************************************************************
```

Starter JCL

```
//ALOGBKUP  JOB
//ALOGBKUP  EXEC  CATALOGB
//SYSIN     DD    *
++DUMP
//
```

CATALOGB Output

This procedure produces the same report (TLMS055) for both the initialization and backup processes. See the *User Guide* for an example of the System Activity Analysis report (TLERPT19).

CATALOGI - Initialize the Alternate Log File

Use CATALOGI to initialize the Alternate Log file. Set the RECOVERY system option in CAI.CTAPOPTN, TLMSIPO to ALTLOG (RECOVRY=ALTLOG). The BLOCKS field and the WARNING field are limited to six digits each.

CATALOGI Procedure JCL

```
//*************************************************************
//**        PROC TO INIT THE CA TLMS  ALTERNATE LOG FILE    *
//*************************************************************
//CATALOGI PROC A='*',
//             ALOG='CAI.TLMS.ALOG',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)'
//*
//INITALOG EXEC PGM=TLMSALOG
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR
//*
//CAIALOG  DD  DSN=&ALOG,DISP=OLD
//*
//SYSUDUMP DD  SYSOUT=&A
//*
//SYSIN    DD  DUMMY
//*
//*************************************************************
```

Starter JCL

```
//ALOGINIT JOB
//ALOGINIT EXEC CATALOGI
//SYSIN DD *
++INIT         BLOCKS=2500,WARNING=2200
//
```

CATALOGI Output

CATALOGI produces the TLMS055 report showing the results of the Alternate Log file initialization.

```
YOUR TLMS NN.n COMPANY NAME              M A I N T E N A N C E  O F  A L T E R N A T E  L O G  F I L E    TLMS055  PAGE  1

CA TLMS NN.n yymmTLrrr                                                                          mm/dd/yyyy  hh.mm.ss



                    CARD INPUT                                                                   ERROR MESSAGES

                    ----------                                                                   --------------



++INIT         BLOCKS=9500,WARNING=9000                                                          CARD ACCEPTED.
```

CATCSMF - Create Volume Master File from SMF Data

This procedure is used to create a VMF using SMF backup data as input. This procedure is used during first installs only.

CATCSMF Procedure JCL

```
//*************************************************************
//*            **** PROCNAME=CATCSMF ****                     *
//*************************************************************
//** PROCEDURE TO CREATE CURRENT CA TLMS VMF FROM SMF DATA  *
//*************************************************************
//** NOTE..THIS PROCEDURE IS FOR INSTALLATION PURPOSES ONLY.*
//*************************************************************
//CATCSMF PROC A='*',
// SPCSMF='(CYL,(10,10))',
// LOAD='CAI.CTAPLINK',
// OPTS='CAI.CTAPOPTN(TLMSIPO)',
// SMFIN='SYS1.MANX',
// SORTLIB='SYS1.SORTLIB',
// VMF='CAI.TLMS.VMF',
// WORK='SYSDA'
//*
//TLMSCSMF EXEC PGM=TLMSCSMF
//*
//STEPLIB DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD DSN=&OPTS,DISP=SHR
//*
//SORTLIB DD DSN=&SORTLIB,DISP=SHR
//*
//SYSPRINT DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT DD SYSOUT=&A
//*
//CAIVMF DD DSN=&VMF,DISP=SHR
//*
//SMFINPUT DD DSN=&SMFIN,DISP=SHR
//*
//SORTWK01 DD UNIT=&WORK,SPACE=&SPCSMF
//*
//SYSUDUMP DD SYSOUT=&A
//*
```

Starter JCL

```
//CSMF     JOB
//SMF2VMF  EXEC CATCSMF
//
```

CATCSMF Output

CATCSMF produces a report (TLMSSMF) listing converted volumes not found in the VMF and their associated data set names.

```
YOUR TLMS NN.n COMPANY NAME              TLMS SMF CONVERSION UTILITY             TLMSSMF PAGE 79

CA TLMS NN.n  yymmTLrrr                                                          hh.mm.ss


400002`UPDATE FAILED 92142/08.05.35 DSN=CAI.IE21.PIMLIB
409873 UPDATE FAILED 92142/08.05.39 DSN=ADMINHR.R30PROD.PAPR0502.G0522V00
409873 UPDATE FAILED 92142/08.14.47 DSN=ADMINHR.R30PROD.PAPR0502.G0522V00
447678 UPDATE FAILED 92142/08.15.10 DSN=SYSQA.AK048BKP.CAICICS
447678 UPDATE FAILED 92142/08.15.33 DSN=SYSQA.AK048BKP.CTAPLINK
447678 UPDATE FAILED 92142/08.15.36 DSN=SYSQA.AK048BKP.CAILPA
447678 UPDATE FAILED 92142/08.15.44 DSN=SYSQA.AK048BKP.CAIMAC
447678 UPDATE FAILED 92142/08.15.47 DSN=SYSQA.AK048BKP.CAIOPTN
447678 UPDATE FAILED 92142/08.15.50 DSN=SYSQA.AK048BKP.CTAPPROC
447678 UPDATE FAILED 92142/08.16.01 DSN=SYSQA.AK048BKP.CTAPSRC
447678 UPDATE FAILED 92142/08.16.43 DSN=SYSQA.AK048BKP.CK048LLD
447678 UPDATE FAILED 92142/08.17.13 DSN=SYSQA.AK048BKP.CK048MLD
447678 UPDATE FAILED 92142/08.17.27 DSN=SYSQA.AK048BKP.CK048PLD
429785 UPDATE FAILED 92142/08.17.43 DSN=SYS2.DB2PM.V120.EXAMPLES
409873 UPDATE FAILED 92142/08.17.43 DSN=ADMINHR.R30PROD.PAPR0502.G0522V00
447678 UPDATE FAILED 92142/08.17.48 DSN=SYSQA.AK048BKP.CK048SLD
447678 UPDATE FAILED 92142/08.17.51 DSN=SYSQA.AK048BKP.CPP10LLD
447678 UPDATE FAILED 92142/08.17.53 DSN=SYSQA.AK048BKP.SMPMTS
447678 UPDATE FAILED 92142/08.17.56 DSN=SYSQA.AK048BKP.SMPPTS
447678 UPDATE FAILED 92142/08.18.00 DSN=SYSQA.AK048BKP.SMPSCDS
447678 UPDATE FAILED 92142/08.18.02 DSN=SYSQA.AK048BKP.SMPSTS
410153 UPDATE FAILED 92142/08.18.58 DSN=ADMINHR.R30PROD.PAPR0502.G0522V00
INX UPDATE FAILED 92142/08.21.39 DSN=INPUT.XTAPE
OUT1X UPDATE FAILED 92142/08.21.39 DSN=OUTPUT.XCART1
OUT2X UPDATE FAILED 92142/08.21.39 DSN=OUTPUT.XCART2
OUT3X UPDATE FAILED 92142/08.21.39 DSN=OUTPUT.XCART3
INX UPDATE FAILED 92142/08.25.19 DSN=INPUT.XTAPE
OUT1X UPDATE FAILED 92142/08.25.19 DSN=OUTPUT.XTAPE1
OUT2X UPDATE FAILED 92142/08.25.19 DSN=OUTPUT.XTAPE2
OUT3X UPDATE FAILED 92142/08.25.20 DSN=OUTPUT.XTAPE3
418983 UPDATE FAILED 92142/08.30.32 DSN=BNKDV.CICSBDVV.CAICSL.G0014V00
410785 UPDATE FAILED 92142/08.33.22 DSN=CAI.CHQA.DS519205.B920502A
418983 UPDATE FAILED 92142/08.47.42 DSN=BNKDV.CICSBDVV.CAICSL.G0014V00
CP8132 UPDATE FAILED 92142/08.50.23 DSN=T1P8132.SMPMCS
CP8132 UPDATE FAILED 92142/08.50.25 DSN=T1P8132.SMPMCS

************ S U M M A R Y ************
TOTAL SMF RECORDS READ 219,323
TOTAL TYPE 14 RECORDS 944
TOTAL TYPE 15 RECORDS 3,226
TOTAL TYPE 21 RECORDS 735
UPDATE FAILURES 4,196
//*******************************************************************
```

CATOPTS - Display TLMS Options

This procedure determines the currently active options.

CATOPTS Procedure JCL

```
//*************************************************************
//*             ****  PROCNAME=CATOPTS  ****                  *
//*************************************************************
//**    PROCEDURE TO DISPLAY THE CA TLMS USER OPTIONS        **
//*************************************************************
//CATOPTS  PROC A='*',
//             CTSOPTS='CAI.CTAPOPTN',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             PRM=ALL
//*
//TLMSSTAT EXEC PGM=TLMSSTAT,PARM='&PRM'
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR
//*
//CTSOPTNS DD  DSN=&CTSOPTS,DISP=SHR
//*
//CTSRPT   DD  SYSOUT=&A
//*
//SYSABEND DD  SYSOUT=&A
//*
```

Starter JCL

```
//TLMSOPTS JOB
//PRTOPTS  EXEC CATOPTS
//
```

CATOPTS Output

```
MONDAY, SEPTEMBER 13, 2010.256                    TLMSSTAT    JOB=JOPTS
  (line continues) STEP=TLMSSTAT    TIME=10:32:58                PAGE=00001
CA TLMS OPTIONS LISTING
    CA TLMS           NN.n           GENLEVEL 104TLC60
GENERAL:
    ALTCTR=D1         COMPANY=CA TLMS DEVELOPMENT
    DATACTR=DC        DATEFMT=MM/DD/YYYY               PAGESIZ=58
    VMFNAM =
    VMFINAM=
    VMFXTND=
    ALT LOG=
TECHNICAL:
    FORSPEC=NO        LOGID=240         LBLDTRY=YES
    QSIZE=064         MSGPFX=CTS        SMS=NO
    NSM=00            SCR=00            EDM=00
    ROUTAUX=NO
    ROUTINQ=YES     ( RT=14       UPD=YES     MSG=DEL                 )
PROCESSING:
    ABEND=24          BRKCHN=OPEN       CATDAYS=1         DISP=OUTPUT
    DBLDRIV=NO        DBLTIME=000000    IDSNVER=YES
    INPUT=YES       ( CHN=NO      BLP=YES     NL=YES      SL=YES       )
    KDATE=MAX         MANUAL=NO         NOTLMS=ABEND      PROTECT=ALL
    RECOVRY=ALTLOG    SCRATCH=CDS       SERIND=YES
    UNCATLG=YES       VSNPAD=           VSNREQD=YES
USER EXITS:
    CLSEXIT=NO        CMDEXIT=NO        EDMEXIT=NO
    NITEXIT=NO        OPNEXIT=NO        SCREXIT=NO
    SECEXIT=TLMSXSEC  SPLEXIT=NO        TRSEXIT=NO
    UPDEXIT=NO
SECURITY:
    PROMPT=NO
    SECURE=NO
TLMSHTAB:
    MSG  ID.          OPTION
    CAT9010P          SUP
    CAT9013P          SUP
TLMSUTAB:
    USER ID. PASSWORD ACCESS
    CRAJO01             U
    ABEWI01             U
```

CATRMFB - Backup the Retention Master File

If the retention master file is being used, a periodic backup of this file, using the CATRMFB procedure should be scheduled. Since tape processing need not be quiesced to perform this procedure. This procedure should also be run before editing the retention master file (RMF).

Process

Prepare the PROC by specifying the RMF and backup data set names. In the TAPE keyword, you must specify the UNIT value for your device. If you wish to backup to DASD, you can specify TAPE='SYSALLDA,SPACE=(TRK,(30,5))'. Finally you need to override the SYSIN DD statement and specify 'REPRO INFILE(CAIRMF) OUTFILE(CAIBRMF)'.

CATRMFB Procedure JCL

The following procedure is used to back up the Retention Master File.

```
//*****************************************************************
//*             ****  PROCNAME=CATRMFB  ****                      *
//*****************************************************************
//*****************************************************************
//**  THIS PROCEDURE USES IDCAMS "REPRO" TO COPY THE VSAM        *
//**  RETENTION MASTER FILE (RMF) TO A BACKUP DATA SET.          *
//**                                                             *
//**  THE SYSIN SHOULD BE: REPRO INFILE(CAIRMF) OUTFILE(CAIBRMF)*
//*****************************************************************
//CATRMFB  PROC A='*',
//             BKUPRMF='CAI.TLMS.BKUPRMF',
//             RMF='CAI.TLMS.RMF',
//             TAPE='TAPE'
//*
//BKUPRMF  EXEC  PGM=IDCAMS
//*
//SYSPRINT DD  SYSOUT=&A
//*
//CAIBRMF  DD  DSN=&BKUPRMF,
//         DSORG=PS,RECFM=FB,LRECL=200,BLKSIZE=0,
//         DISP=(NEW,CATLG),UNIT=&TAPE
//*
//CAIRMF   DD  DSN=&RMF,
//         DISP=OLD
//*
//SYSIN    DD  DUMMY
//*
```

Starter JCL

```
//RMFBKUP  JOB
//RMFBKUP  EXEC  CATRMFB
//             BKUPRMF='CAI.TLMS.BKUPRMF',
//             RMF='CAI.TLMS.RMF'
```

CATRMFB Output

This procedure produces an IDCAMS listing.

CATRMFE – Update Pattern Masked RMF

This procedure is used to update an RMF which contains pattern masked retention rules. This procedure invokes TLMSRMFE which allow adding and deleting retention locations and retention rules. It also will initialize a pattern masked RMF or will convert a non-pattern masked RMF into a pattern masked RMF.

Process

Add a SYSIN DD statement to the job stream.

CATRMFE Procedure JCL

The following procedure is used to update the Retention Master File.

```
//*************************************************************
//*             ****  PROCNAME=CATRMFE  ****                  *
//*************************************************************
//*     PROCEDURE TO UPDATE CA TLMS TAPE RETENTION MASTER FILE *
//*               FOR PATTERN MASKING RMF ONLY                 *
//*************************************************************
//CATRMFE  PROC A='*',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             PRM=' ',
//             RMF='CAI.TLMS.RMF'
//*
//*
//UPDTRMF  EXEC PGM=TLMSRMFE,
//             PARM='&PRM'
//*
//*********  PRM='1ST,2ND,3RD,4TH,5TH **************************
//* WHERE  1ST = NUMBER OF ENTRIES TO BE CONVERTED        2000 *
//*        2ND = NUMBER OF UPDATES                         500 *
//*        3RD = NUMBER OF CABINETS (MAX FOR ONE LOCATION) 160 *
//*        4TH = NUMBER OF BOXES                           104 *
//*        5TH = NUMBER OF LOCATION RECORDS               1024 *
//*        OMITTED PARM TAKE DEFAULTS   PRM=',825,,50'         *
//*************************************************************
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR     OPTIONS WHEN TLMS IS DOWN
//*
//CAIRMF   DD  DSN=&RMF,DISP=OLD           RETENTION MASTER FILE
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A   REPORTS
//*
//SYSUDUMP DD  SYSOUT=&A                   ABEND DUMPS
//*
//SYSIN    DD  DUMMY                       OVERRIDE FOR COMMANDS
//*
//*************************************************************
```

Starter JCL

```
//RMFINIT JOB
//DEFINE   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD   SYSOUT=*
//SYSIN DD *
 DELETE CAI.RMF
 DEFINE CLUSTER                      -
        (NAME (CAI.RMF)              -
         VOL(VSN010)                 -
         RECSZ(200 200)              -
         SHR(3 3) INDEXED REPL       -
         IMBED KEYS(52 1))           -
         DATA                        -
         (NAME(CAI.RMF.DATA)         -
         FREESPACE(40 30)            -
         TRK(3 3))                   -
         INDEX                       -
         (NAME(CAI.RMF.INDEX)        -
         TRK(1 1))
/*
//*
//RMFINIT EXEC CATRMFE
//UPDTRMF.SYSIN DD *
/*********************************/
/* INITIALIZE PM RMF */
/*********************************/
INIT
//
```

CATRMFE Output

CATRMFE produces the TLMS054 report showing the results of the update of the pattern masked RMF.

```
MYCOMPANY                                RETENTION MASTER FILE UPDATE REPORT                TLMS054    PAGE      1
 CA TLMS        NN.n    103TLC60                                                                      10-09-2008    08.13.58
  STMT NO                                  SYSIN DATA                                                PARSE ERRORS
   ------- ------------------------------------------------------------------------------  ----------------------------------------

           /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
           /* THIS IS A COMMENT BOX. THEY MAY APPEAR ANYWHERE                         */
           /*                                                                         */
           /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

      0001   ADDLOC ID(DC) DESC(DATA CENTER          (DC))                      +
             BOXLIST(BOXA10 BOXB10 BOXC10 BOXD10 BOXE10 BOXF10 BOXG10 BOXH10 BOXI10) +
             BOXLIST(BOXJ10 BOXK10 BOXL10 BOXM10 BOXN10 BOXO10 BOXP10 BOXQ10 BOXR10) +
             BOXLIST(BOXS10 BOXT10 BOXU10 BOXV10 BOXW10 BOXX10 BOXY10 BOXZ10)


      0002   ADDLOC ID(AR) DESC(ARCHIVE BACKUPS      (AR))                      +
                  CABLIST(A199 A299 KA39 A499 A599 A699 A799 A899 A999 )


      0003   ADDLOC ID(BL) DESC(BOB'S TEST LOCATION  (BL))                      +
                  CABLIST(AA99 AB49 AC49 AD49 AE49 AF99 AG99 AH49 AI49 AJ49) +
                  CABLIST(AK49 AL49 AM49 AN49 AO05 AP05 AQ05 AR05 AS05 AT05) +
                  CABLIST(BA99 BB49 BC49 BD49 BE49 BF99 BG99 BH49 BI49 BJ49) +
                  CABLIST(BK49 BL49 BM49 BN49 BO05 BP05 BQ05 BR05 BS05 BT05) +
                  CABLIST(CA99 CB49 CC49 CD49 CE49 CF99 CG99 CH49 CI49 CJ49) +
                  CABLIST(CK49 CL49 CM49 CN49 CO05 CP05 CQ05 CR05 CS05 CT05) +
                  CABLIST(DA99 DB49 DC49 DD49 DE49 DF99 DG99 DH49 DI49 DJ49) +
                  CABLIST(DK49 DL49 DM49 DN49 DO05 DP05 DQ05 DR05 DS05 DT05)


      0004   ADDLOC ID(B1) DESC(BOX LOCATAION 1      (B1)) BOXLIST()


      0005   ADDLOC ID(B2) DESC(BOX LOCATAION 2      (B2)) BOXLIST()


      0006   ADDLOC ID(D1) DESC(ALTERNATE DATA CENTER(D1))


      0007   ADDLOC ID(KP) DESC(KEEP FOREVER         (KP)) +
                  CABLIST(KA99 KB99 KC99 KD99 KE99 KF99 KG99               )

      0008   ADDLOC ID(L1) DESC(OFFSITE STAGE 1      (L1)) +
                  CABLIST(AA99 AB49 AC49 AD49 AE49 AF99 AG99 AH49 AI49 AK49)

      0010   ADDLOC ID(TX) DESC(TRANSMISSIN DEPT     (TX))

      0011   ADDLOC ID(ZZ) DESC(TEST LOCATION        (ZZ))

      0012   ADDRTN DSN(DEFAULT)                             OWNER(AUTHORITY) +
                    RTN(2DC )

      0013   ADDRTN DSN(ASM2-)                               OWNER(AUTHORITY) +
                    RTN(6DC 7AR)



          CAT3318I RMF UPDATES STARTED
          CAT3319I RMF UPDATES COMPLETE
          CAT3308I END OF RMF UPDATE REPORT    RC = 00
```

CATRMFI - Initialize the Retention Master File

This procedure is used to initialize and allocate the RMF.

CATRMFI Procedure JCL

```
//RMFINIT  JOB
//DEFINE   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DELETE CAI.RMF
  DEFINE CLUSTER                      -
         (NAME (CAI.RMF)              -
          VOL(ASM011)                 -
          RECSZ(200 200)              -
          SHR(3 3) INDEXED REPL       -
          IMBED KEYS(52 1))           -
          DATA                        -
          (NAME(CAI.RMF.DATA)         -
          FREESPACE(40 30)            -
          TRK(3 3))                   -
          INDEX                       -
          (NAME(CAI.RMF.INDEX) -
          TRK(1 1))
/*
//*
//RMFINIT  EXEC CATRMFE
//RMFINIT.SYSIN DD *
  /* * * * * * * * * * * * * * */
  /*   INITIALIZE THE RMF    */
  /* * * * * * * * * * * * * * */
  INIT
/*
//
//RMFCONV  EXEC CATRMFE
//RMFCONV.SYSIN DD *
  /* * * * * * * * * * * * * * */
  /* CONVERT OLD RMF TO PM RMF */
  /* * * * * * * * * * * * * * */
  CONVERT
/*
//
//RMFUPDT  EXEC CATRMFE
//RMFUPDT.SYSIN DD *
  /* * * * * * * * * * * * * * */
  /* INITIAL UPDATE OF  PM RMF */
  /* * * * * * * * * * * * * * */
  ADDLOC ID(DC)  DESC(DATA CENTER (DC) )
  ADDRTN DSN(DEFAULT)               OWNER(TAPE LIBRARIAN) +
         RTN(2DC )
/*
//
```

CATRMFI Output

CATRMFI produces report TLMS051, which lists the RMF initialization control statements were processed successfully, and the number of records which were reserved for each index.

```
YOUR TLMS COMPANY NAME                  FORMAT RETENTION MASTER                                    PAGE      1
 CA TLMS        NN.n    yymmTLrrr                                                          mm-dd-yyyy    hh.mm.ss


                                    ADDITIONS TO RETENTION MASTER FILE
                                    ----------------------------------
                                    QUALIFIER        NUMBER OF RECORDS
                                    ---------        -----------------
                                    XYZ                    1

                                           END OF JOB
```

CATRMFP– Reports for Pattern Masked RMF

This procedure is used to print one or more RMF reports. The TLMS014 report list the available cabinet/slots and boxes within locations. The TLMS015 report list the locations definitions. The TLMS016 report list the retention rules.

Process

Override the SYSIN DD statement and add statements for the reports you wish printed.

CATRMFP Procedure JCL

The following procedure is used to produce reports from the Retention Master File.

```
//*************************************************************
//*             ****  PROCNAME=CATRMFP  ****                  *
//*************************************************************
//*          PROCEDURE TO PRINT CA TLMS RETENTION MASTER FILE *
//*************************************************************
//CATRMFP  PROC A='*',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             PRM=' ',
//             RMF='CAI.TLMS.RMF',
//             RMF='CAI.TLMS.VMF'
//*
//*
//REPTRMF  EXEC PGM=TLMSRMFP,PARM='&PRM'
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR   OPTIONS WHEN TLMS IS DOWN
//*
//CAIRMF   DD  DSN=&RMF,DISP=OLD    RETENTION MASTER FILE
//*
//CAIVMF   DD  DSN=&VMF,DISP=SHR    VOLUME MASTER FILE(TLMS014)
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A  REPORTS
//*
//SYSUDUMP DD  SYSOUT=&A                  ABEND DUMPS
//*
//SYSIN    DD  DUMMY                      OVERRIDE FOR COMMANDS
//*
//*************************************************************
```

Starter JCL

```
//RMFPRINT   JOB
//RMFPRINT   EXEC CATRMFP
//REPTRMF.SYSIN DD *
TLMS014
TLMS015
TLMS016
//
```

CATRMFP Output

CATRMFP produces the TLMS014, TLMS015, and TLMS016 reports showing the results of the update of the pattern masked RMF.

```
1XE90 TLMS DEVELOPMENT                              REPORT REQUEST                                                  PAGE      1
 CA TLMS        NN.n    103TLC60                                                                  10-09-2008    08.14.34
-PAGESIZE=58
 DATE=MM-DD-YYYY
-TLMS016
1XE90 TLMS DEVELOPMENT                              TAPE RETENTION SELECTION RECORDS                 TLMS016    PAGE      1
 CA TLMS        NN.n    1003TLC00                                                                 10-09-2008    08.14.34
-
    TYPE     DATASET PATTERN MASK                                                  JOB PATTERN MASK              ADD   DATE
     -       ------------------------------------------------------------------    ----------------              ----------
         DSN=**DEFAULT**                                                                                         10-09-2008
             RTN(2DC                                    )          OWNER(AUTHORITY                )
     5   DSN=ASM2-                                                                 JOB=*                         10-09-2008
             RTN(6DC     7AR                            )          OWNER(AUTHORITY                )
     5   DSN=BOXIT-                                                                JOB=*                         10-09-2008
             RTN(5DC0001 5B10005                        )          OWNER(AUTHORITY                )
     5   DSN=CATAPE.-                                                              JOB=*                         10-09-2008
             RTN(5DC0001 7KP                            )          OWNER(AUTHORITY                )
     5   DSN=CRAJO01.-                                                             JOB=*                         10-09-2008
             RTN(1DC     5B20005                        )          OWNER(AUTHORITY                )
     5   DSN=DASTER.-                                                              JOB=*                         10-09-2008
             RTN(6DC     7KP                            )          OWNER(AUTHORITY                )
     5   DSN=FRED.-                                                                JOB=*                         10-09-2008
             RTN(1DC     7ZZ                            )          OWNER(AUTHORITY                )
     5   DSN=HSM-                                                                  JOB=*                         10-09-2008
             RTN(6DC     7AR                            )          OWNER(AUTHORITY                )
     0   DSN=MGMT0000                                                              JOB=*                         10-09-2008
             RTN(6DC     7L1                            )          OWNER(AUTHORITY                )
     5   DSN=PROD1-                                                                JOB=*                         10-09-2008
             RTN(6DC     5L10007                        )          OWNER(AUTHORITY                )
     5   DSN=PROD2-                                                                JOB=*                         10-09-2008
             RTN(6DC     5L20007                        )          OWNER(AUTHORITY                )
     5   DSN=PROD3-                                                                JOB=*                         10-09-2008
             RTN(6DC     5AR0007                        )          OWNER(AUTHORITY                )
     5   DSN=S-                                                                    JOB=*                         10-09-2008
             RTN(6DC     5L10017 5L20001 5AR0030 7KP    )          OWNER(AUTHORITY                )
     5   DSN=TEST-                                                                 JOB=*                         10-09-2008
             RTN(1DC     2ZZ0018                        )          OWNER(AUTHORITY                )
     5   DSN=TST1-                                                                 JOB=*                         10-09-2008
             RTN(8DC0003 AL1                            )          OWNER(AUTHORITY                )
     5   DSN=TST2-                                                                 JOB=*                         10-09-2008
             RTN(8DC0003 3L10020                        )          OWNER(AUTHORITY                )
     0   DSN=UNIX.MASTER.TAPE.DUP                                                  JOB=WILRO09                   10-09-2008
             RTN(2DC0007 4L20023                        )          OWNER(AUTHORITY                )
     0   DSN=UNIX.MASTER.TAPE.DUP                                                  JOB=*                         10-09-2008
             RTN(6DC     5L10021 7KP                    )          OWNER(AUTHORITY                )
1XE90 TLMS DEVELOPMENT                              TAPE RETENTION SELECTION RECORDS                 TLMS016    PAGE      2
 CA TLMS        NN.n    yymmTLrrr                                                                 10-09-2014    08.14.34


                 ***************************************************
                 *                  L E G E N D                    *
                 *                                                 *
                 *     TYPE      DESCRIPTION                       *
                 *       -       ------------------------------    *
                 *       0       SPECIFIC                          *
                 *       1       RESERVED FOR SPECIAL SEARCH       *
                 *       2       ONE-FOR-ONE MASK (# @ !)          *
                 *       3       PREFIXED/CONTAINING/SUFFIXED      *
                 *       4       PREFIXED/SUFFIXED                 *
                 *       5       PREFIXED ONLY                     *
                 *       6       RESERVED FOR SPECIAL SEARCH       *
                 *       7       SUFFIXED ONLY                     *
                 *       8       CONTAINING ONLY                   *
                 *       9       MAX ITERNAL CLASS VALUE           *
                 *                                                 *
                 ***************************************************
```

CATVMFB - Back Up the Volume Master File

The TLMSVMFU program (CATVMFB procedure) is used to back up the Volume Master File. Periodic backup of the VMF is necessary to aid in its recovery should the VMF be corrupted or lost. The backup procedure is also used to save transactions which occurred since the last backup was performed. The following should be considered when scheduling this function:

- Tape processing does not have to be quiesced to perform a VMF backup procedure.
- The level of tape activity in the data center may determine how often the VMF should be backed up. A high level of activity indicates the necessity for frequent backups.

CATVMFB Procedure JCL

The following procedure is used to back up the Volume Master File. BUFNO is supplied to speed the reading of an unblocked VMF. Choose a size which is a factor or multiple of the number of blocks per track, for example, 48 for a 3380 device.

```
//*********************************************************
//*             **** PROCNAME=CATVMFB ***                 *
//*********************************************************
//** PROCEDURE TO BACK UP CA TLMS VOLUME MASTER FILE     *
//*********************************************************
//CATVMFB PROC A='*',
// BKUPVMF='CAI.TLMS.BKUPVMF',
// BLKVMFB='8000',
// BUFNO='80',
// LOAD='CAI.CTAPLINK',
// OPTS='CAI.CTAPOPTN(TLMSIPO)',
// TAPE='TAPE',
// VMF='CAI.TLMS.VMF'
//*
//*********************************************************
//** EXECUTE VOLUME MASTER FILE UTILITY                  *
//*********************************************************
//BKUPVMF EXEC PGM=TLMSVMFU,PARM='++DUMP'
//*
//STEPLIB DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD DSN=&OPTS,DISP=SHR
//*
//CAIVMF DD DSN=&VMF,DISP=SHR
//*
//CAIVMFS DD DSN=&VMF,DISP=SHR,
// DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIBVMF DD DSN=&BKUPVMF,
// DISP=(NEW,CATLG,DELETE),
// DCB=(DSORG=PS,LRECL=500,BLKSIZE=&BLKVMFB,RECFM=FB),
// UNIT=&TAPE
//*
//SYSPRINT DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT DD SYSOUT=&A
//*
//SYSUDUMP DD SYSOUT=&A
//*
//*********************************************************
```

Starter JCL

```
//VMFBKUP JOB
//VMFBKUP EXEC CATVMFB
//
```

CATVMFB Output

The ++DUMP parameter produces the Dump Volume Master Report (TLMS048).

```
YOUR TLMS COMPANY NAME                VOLUME MASTER FILE UTILITY FOR ++DUMP              TLMS048    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                   mm/dd/yyyy hh.mm.ss
                                      TOTAL INPUT RECORDS READ...................615
                                      TOTAL CONTROL RECORDS WRITTEN..............1
                                      TOTAL SEGMENT RECORDS WRITTEN..............1
                                      TOTAL BASE RECORDS WRITTEN.................461
                                      TOTAL MVL RECORDS WRITTEN..................0
                                      TOTAL MDS RECORDS WRITTEN..................12
                                      TOTAL AUXILIARY RECORDS WRITTEN............90
                                      TOTAL MESSAGE RECORDS WRITTEN..............50
                                      TOTAL RECORDS WRITTEN......................615
                                      TOTAL ORPHANED MVL RECORDS.................0
                                      TOTAL ORPHANED MDS RECORDS.................0
                                      TOTAL ORPHANED VSN RECORDS.................0
                                      TOTAL WARNING MESSAGES.....................0
```

TOTAL INPUT RECORDS READ

Total records contained in the VMF backup (control, segment, base, MVL, MDS, message and auxiliary).

TOTAL CONTROL RECORDS WRITTEN

Number of control records written to the VMF.

TOTAL SEGMENT RECORDS WRITTEN

Number of segment records written to the VMF.

TOTAL BASE RECORDS WRITTEN

Number of volume records written to the VMF.

TOTAL MVL RECORDS WRITTEN

Number of multivolume records written to the VMF.

TOTAL MDS RECORDS WRITTEN

Number of auxiliary records used to chain multi-data set volumes.

TOTAL AUXILIARY RECORDS WRITTEN

Number of unused auxiliary records left in VMF for chaining.

TOTAL MESSAGE RECORDS WRITTEN

Number of message records written to the VMF.

TOTAL RECORDS WRITTEN

Total of all records written to the VMF (control, segment, base, MVL, MDS, message and auxiliary).

TOTAL ORPHANED MVL RECORDS

This total reflects the number of unchained multivolume records that were read and processed as auxiliary records.

TOTAL ORPHANED MDS RECORDS

This total reflects the number of unchained multi-data set records that were read and processed as auxiliary records.

TOTAL ORPHANED VSN RECORDS

This total reflects the VSN records which were not copied from a source VMF because they were not defined by the IDCK.

TOTAL WARNING MESSAGES

This is the total number of warning messages written.

The following describes the fields in all reports produced by TLMSVMFU.

CHAINING RECORDS RESERVED

The total number of records reserved for MVL and MDS records.

MESSAGE RECORDS RESERVED

The number of records reserved for auxiliary messages.

CHAINING THRESHOLD COUNT

The number of remaining chaining records when WARNING message should be issued.

VOLUME INITIALIZED AS

Reflects which volumes are initialized as SCRATCH OR NOSCRATCH.

COMPANY

Company name in VMF.

CATVMFI - Initialize the Volume Master File

CA TLMS requires a BDAM Volume Master File which contains one 500-byte record for each tape volume in the tape library under the control of CA TLMS (including standard labeled, nonlabeled, bypass label processing and nonstandard labeled tape volumes). CA TLMS controls all tapes by volume serial number (VSN).

Access to the VMF is by the relative record BDAM technique. The relative record is derived from the volume serial number of the tape volume. This file may be blocked. Since tape VSNs are not restricted to a single consecutive numeric series, direct-access storage is conserved by using a file segmentation technique. Records exist in the VMF for all consecutive numbers within a file segment.

CA TLMS maintains an index of file segments in the Volume Master File in one or more segment records at the beginning of the file. One segment record can hold up to 23 file segment indexes. The Volume Master File will contain as many segment records as required to hold the file segment indexes.

A small number of skipped or missing VSNs does not require the definition of a new file segment. These numbers are defined as skipped when the file is built and dummy records are written to the file to maintain the relative record logic.

The VMF also contains records that are used to maintain multivolume data sets and multi-data set volumes. These are defined as auxiliary records. You can also reserve additional space for special disposition messages which are generated when specific tape data sets are closed.

Process

The TLMSVMFU program is used to initialize the VMF. It creates

- A control record for internal usage
- Segment records to define the specific volume serial number ranges in the database
- Auxiliary records to handle data sets that span more than one volume and volumes that contain more than one data set
- Optional message records which allow you to provide additional information when a tape volume is closed

Prepare the initialization commands for input into the Volume Master File initialization program and save in CAI.CTAPOPTN member TLMSIDCK. The commands are free-form. These commands are described on the following pages.

Note: For information on preparing these commands, see Volume Master File Allocation Worksheet (see page 227).

Initialization Commands

There are three types of statements in the TLMSIDCK member:

- CONTROL

  Describe various controls for the VMF

- RANGE

  Defines the VOLSER ranges

- SKIP

  Defines unused spaces in ranges

The following keywords may appear on one or more CONTROL statements.

This parameter is required. The COMPANY name must be enclosed in single quotes.

AUX=

Reserve space for auxiliary records. Auxiliary records keep track of volumes that contain more than one data set and data sets that span more than one volume. Auxiliary space is managed so that available records are reused without compressing the VMF. This parameter is required. The minimum number of auxiliary records is 100.

The descriptions below help you with your estimates.

- Multi-data set volumes: If a volume contains more than one data set, the base record holds the first data set and each auxiliary record holds a maximum of two additional data sets. If a volume contains more than three data sets, additional auxiliary records are chained together until all of the data sets are recorded. For example, five data sets on the same volume would be recorded in one base record and two auxiliary records, like this:

VOLABC base record	VOLABC auxiliary record 1	VOLABC auxiliary record 2
DSN1	DSN2	DSN4
	DSN3	DSN5

- Multivolume data sets: If a data set spans more than one volume, the base record holds the base volume and up to five additional volumes. Each auxiliary record holds up to 32 additional volumes. If a data set requires more than 38 volumes, additional auxiliary records are chained together until all the volumes are recorded. For example, 70 volumes would be listed in one base record and two auxiliary records, like this:

VOL001 base record	VOL001 auxiliary record 1	VOL001 auxiliary record 2
VOL002	VOL007	VOL039
VOL003	VOL008	VOL040
VOL004	VOL009	VOL041
VOL005	-	-
VOL006	-	-
	-	-
	VOL036	VOL068
	VOL037	VOL069
	VOL038	VOL070

COMPANY=

Provide a company name (1-30 characters) which will be printed on each CA TLMS report.

CPU=

Assign a CA TLMS internal system identification character to each SMF system identification symbol.

A maximum of 16 CPUs may be specified, 8 per CONTROL statement.

MESSAGE=

Indicate whether you want to reserve 50 records for auxiliary disposition messages (Y or N). This parameter is required.

SCRATCH=

Indicate whether all volume records in the VMF will be initialized as scratch volumes: Y (scratch) or N (nonscratch). This parameter is required.

WARNING=

Define when to generate warning messages that indicate that the auxiliary records area is almost full. Specify a number that is less than the number specified above for AUX. For example, if AUX is 5000, specify 4000 for WARNING. You will receive a warning message when 80 percent of the auxiliary messages have been used. This field must be specified on the same statement as AUX, separated by a comma.

The second type of statement in the TLMSIDCK member is the RANGE statement. At least one RANGE statement must be included, and the number of ranges is limited only by VMF size and performance. The format is as follows:

```
RANGE VSN(bbbbbb eeeeee) LEN(llll) DATE(jjjjj) MFG(mmmmmmmm) +

      TYPE(tt) ATL(ttt nnn) DEN(dddd)
```

RANGE

Specifies statement identifier and must appear first on the statement.

VSN( )

Contains the beginning and ending VSN of the range. The ending VSN may be omitted for a range of one tape. The beginning and ending VSN range must be alphanumeric.

LEN

(Optional) Specifies tape length.

DATE

(Optional) Specifies the Julian date recorded in the VMF purchase date.

MFG

(Optional) Specifies to supply the value of the vendor field in the VMF.

TYPE

(Optional) Specifies to supply the value of the tape type field of the VMF. This is a user determined value.

ATL

(Optional) Allows information about the ATL to be stored in the VMF for a range of tapes when it is defined. As with the other range keywords, these values only apply to newly-defined tape volumes and do not change the value of records during ++RESTORE or ++REORG. The ATL keyword is useful for defining virtual tape ranges, and new ranges of tapes added to an ATL. There are two positional sub-parameters associated with ATL.

ATL Type

Specifies a 1 to 8 character CA defined name for type of the automatic tape library. If the library contains virtual tapes, the first character of the name is always V. For example, IBM, VIBM, VTAPE, STORTEK, and VEMC.

ATL Number

Used to distinguish between multiple ATLs of the same type. It can have a numeric value of 1 to 254.

DEN

(Optional) Specifies the default tape density. This value will be overridden when a tape file is opened.

The third type of statement in the TLMSIDCK member is the SKIP statement. This statement is optional and may be included to define some numbers of VSNs within a range which are to be skipped. Thus you can avoid creating additional ranges just to skip a few VSNs. Performance is degraded by large numbers of ranges.

```
SKIP VSN(bbbbbb eeeeee)
```

SKIP

Specifies the statement identifier and must appear first on the statement.

VSN( )

Contains the beginning and ending VSNs to be skipped. The ending VSN may be omitted for a range of one tape.

```
/************************************************/
/* SAMPLE VMF INIT-DECK FOR TLMS TEST SYSTEM        /
/************************************************/
CONTROL COMPANY=(YOUR TLMS RNN.n COMPANY NAME )
CONTROL AUX=00500 WARNING=000450 MESSAGE=Y SCRATCH=Y
CONTROL CPU=(XE90/A SYS2/B SYS3/C DUMY/X )
/*PRINTCPU= */
RANGE VSN(000001 000002) LEN(2400)
RANGE VSN(400001 400050)  DATE(96205)
RANGE VSN(980001 980050)   MFG(MEMOREX)
RANGE VSN(ARC001 ARC020) DATE(96315) LEN(9000) MFG(IBM)
RANGE VSN(HSM001 HSM020)
RANGE VSN(ODD020 ODD073)
RANGE VSN(XMIT50 XMIT60)
RANGE VSN(TST001 TST050)
RANGE VSN(TRN001 TRN010)
SKIP  VSN(980027 980027)
SKIP  VSN(980040 980043)
SKIP  VSN(ODD021 ODD025)
SKIP  VSN(ODD027 ODD069)
SKIP  VSN(ODD070 ODD072)
```

CATVMFI Procedure JCL

The following procedure is used to initialize the Volume Master File.

```
//************************************************************
//* ****                 PROCNAME=CATVMFI ****                *
//************************************************************
//************************************************************
//**  PROCEDURE TO INITIALIZE THE CA TLMS VOLUME MASTER FILE *
//************************************************************
//CATVMFI PROC A='*',
//           BUFNO='80',
//           IDCK='CAI.CTAPOPTN(TLMSIDCK)',
//           LOAD='CAI.CTAPLINK',
//           OPTS='CAI.CTAPOPTN(TLMSIPO)',
//           VMF='CAI.TLMS.VMF'
//*
//************************************************************
//**     EXECUTE VOLUME MASTER FILE UTILITY                 *
//************************************************************
//INITVMF    EXEC PGM=TLMSVMFU,PARM='++INIT'
//*
//STEPLIB    DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS   DD DSN=&OPTS,DISP=SHR
//*
//CAIVMFS    DD DSN=&VMF,DISP=OLD,
//              DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIIDCK    DD DSN=&IDCK,DISP=SHR
//*
//SYSPRINT   DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSUDUMP   DD SYSOUT=&A
//************************************************************
```

Starter JCL

```
//VMFINIT JOB
//VMFINIT EXEC CATVMFI
```

CATVMFI Output

The VMF initialization program (TLMSVMFU) produces the Initialize Volume Master File report (TLMS045). This report details the file segments that were created, their volume serial number ranges, initialized data fields (if any), and skipped VSN ranges. It also summarizes the total record counts for the VMF.

```
YOUR TLMS COMPANY NAME                VOLUME MASTER FILE UTILITY FOR ++INIT              TLMS045    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                   mm/dd/yyyy  hh.mm.ss

                                      CONTROLDATA
                                               ---------------------------------------------
                                               CHAINING RECORDS RESERVED.....................1,234
                                               MESSAGE RECORDS RESERVED.......................50
                                               CHAINING THRESHOLD COUNT.....................1,000
                                               VOLUMES INITIALIZED AS.................SCRATCH
                                               COMPANY.........HANDY VMF BUILDERS, INC

                                               TOTAL INPUT RECORDS READ.....................4,637
                                               TOTAL CONTROL RECORDS WRITTEN.................1
                                               TOTAL SEGMENT RECORDS WRITTEN.................1
                                               TOTAL BASE RECORDS WRITTEN................3,351
                                               TOTAL MVL RECORDS WRITTEN.....................0
                                               TOTAL MDS RECORDS WRITTEN.....................0
                                               TOTAL AUXILIARY RECORDS WRITTEN.........1,234
                                               TOTAL MESSAGE RECORDS WRITTEN................50
                                               TOTAL RECORDS WRITTEN.....................4,637
                                               TOTAL ORPHANED MVL RECORDS....................0
                                               TOTAL ORPHANED MDS RECORDS....................0
                                               TOTAL ORPHANED VSN RECORDS....................0
                                               TOTAL WARNING MESSAGES........................0
```

Note: For information on the fields in this report, see the section CATVMFB Output (see page 218).

Volume Master File Allocation Worksheet

Use this worksheet to estimate the amount of direct-access storage you will need for the VMF.

Note: Do not allocate the VMF and ALOG on the same volume. This is for recovery support purposes. The ALOG is needed for recovery if the volume containing the VMF fails. Do not allocate the VMF, ALOG, or VMFINDEX files on a volume that contains a USER CATALOG. This may cause performance issues. The VMF, VMFINDEX, and RMF may be on the same volume.

Number of tape volumes are in your current tape library:

VOLTOT = _____

Expected tape library growth in the next 12 months as a percentage: (If negative, enter 0.)

PCTGROW = _____

Percentage of data sets that span more than six volumes:

PCTMV = _____

Percentage of tape volumes that contain more than one data set:

PCTMD =_____

Average number of data sets on a multi-data set volume:

AVGMD = _____

Do you plan to use auxiliary messages? (If yes, enter 1. If no, enter 0.)

AUXMSG = _____

Number of records per cylinder for the device CA TLMS VMF is stored on:

RPCYL = _____

Using the values you have supplied, perform the following calculations:

Calculate additional records:

ADDRECS = VOLTOT * PCTGROW/100

ADDRECS = _____

Calculate multivolume record count:

MVRECS = VOLTOT * PCTMV/100

MVRECS = _____

Calculate multi-data set record count:

MDRECS = (VOLTOT * PCTMD/100)/2

MDRECS = _____

Calculate number of records:

TOTRECS = VOLTOT + ADDRECS + MVRECS + (MDRECS * AVGMD) + (AUXMSG * 50)

TOTRECS = _____

Calculate number of cylinders:

TOTCYLS = TOTRECS * 1.5 / RPCYL + 0.5

TOTCYLS = INTEGER _____

Code your JCL SPACE parameter as:

SPACE = CYL,(TOTCYLS)

SPACE = _____

CATVMFID - Generate VMF Initialization Parameters

In the event that the initialization parameters used as input to the CATVMFI procedure are lost, you may execute the CATVMFID procedure to rebuild them based on the current VMF.

This procedure invokes program TLMSVMFU and reads the VMF sequentially from the CAIVMF DD. It then generates the initialization parameters and reproduces them. MFG, TYPE, LEN, and DATE fields on the RANGE statement are not generated.

CATVMFID Procedure JCL

The following procedure is used to generate the parameters used to initialize the Volume Master File.

```
//*************************************************************
//*         **** PROCNAME=CATVMFID ****                       *
//*************************************************************
//** PROCEDURE TO GENERATE THE VMF INITIALIZATION PARAMETERS *
//*************************************************************
//CATVMFID PROC A='*',
//    BUFNO='80',
//    LOAD='CAI.CTAPLINK',
//    OPTS='CAI.CTAPOPTN(TLMSIPO)',
//    VMF='CAI.TLMS.VMF'
//*
//*************************************************************
//**    EXECUTE VOLUME MASTER FILE UTILITY                   *
//*************************************************************
//INITDECK EXEC PGM=TLMSVMFU,PARM='++INITDECK '
//*
//STEPLIB DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD DSN=&OPTS,DISP=SHR
//*
//CAIVMFS DD DSN=&VMF,DISP=SHR,
//      DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//SYSPRINT DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSUDUMP DD SYSOUT=&A
//*
//*************************************************************
```

Starter JCL

```
//VMFDECK JOB
//VMFDECK EXEC CATVMFID
//
```

CATVMFID Output

The ++INITDECK parameter produces a report (TLMS046) listing the control statements that were used to initialize the VMF.

```
YOUR TLMS COMPANY NAME                 TLMS CONTROL CARDS USED TO INITIALIZE THE VMF                TLMS046    PAGE       1
CA TLMS NN.n yymmTLrrr                                                                              mm/dd/yyyy  hh.mm.ss
THE FOLLOWING PARAMETERS WERE USED TO INITIALIZE THE VMF:
CONTROL COMPANY=(TLMS TEST SYSTEM - TLC00 )
CONTROL AUX=000100 WARNING=000070 MESSAGE=Y SCRATCH=Y
CONTROL CPU=(XE90/A SYS2/B SYS3/C SYS4/E
/* PRINTCPU= */
RANGE               VSN(980001 980100)
RANGE               VSN(ASM201 ASM220)
RANGE               VSN(A6A001 A6A020)
RANGE               VSN(NL0050 NL0080)
RANGE               VSN(TDI001 TDI099)
RANGE               VSN(TST001 TST040)
RANGE               VSN(ZZ0100 ZZ0250)
SKIP                VSN(980027 980027)
SKIP                VSN(980040 980043)


                            VMFFILE VMF BASE SKIPPED
SEGMENTS RECORDS RECORDS
TOTALS: 7 461 5
```

The parameters generated on the report may be extracted and then used as input to the CATVMFI procedure to initialize the Volume Master File.

CATVMFMR - Merge Volume Master Files

This process will merge the backup files of two VMFs into a new VMF defined in the TLMSIDCK member in the CAI.CTAPOPTN data sets. VOLSERs not defined in TLMSIDCK are not merged, and VOLSERs defined by TLMSIDCK but not in either VMF backup are generated. If a VOLSER is contained in both backups and the volser(s) is in the TLMSIDCK, the program stops with an error. MDS and MVL records for volumes not merged are ignored, and reported as orphans.

CATVMFMR Procedure JCL

The following procedure is used to merge the backups of two VMFs into a new VMF.

```
//*******************************************************
//* ****          PROCNAME=CATVMFMR                *****
//*******************************************************
//** PROCEDURE TO MERGE TWO VOLUME MASTER FILES        *
//*******************************************************
//* Comments
//CATVMFMR PROC A='*',
// BKUPVMF='CAI.TLMS.BKUPVMF', (VMF - A)
// BUFNO='80',
// IDCK='CAI.CTAPOPTN(TLMSIDCK)',
// LOAD='CAI.CTAPLINK',
// OPTS='CAI.CTAPOPTN(TLMSIPO)',
// MERGE='CAI.TLMS.MRGE.VMF', (VMF - B)
// SORTLIB='SYS1.SORTLIB',
// SPCVMF='(CYL,(5,5))',
// WORK='SYSDA',
// VMF='CAI.TLMS.VMF' (New merged VMF file)
//*
```

```
//*************************************************
//**    EXECUTE VOLUME MASTER FILE UTILITY         **
//*************************************************
//MRGEVMF EXEC PGM=TLMSVMFU,PARM='++MERGE'
//*
//STEPLIB DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD DSN=&OPTS,DISP=SHR
//*
//SORTLIB DD DSN=&SORTLIB,DISP=SHR
//*
//CAIVMF DD DSN=&VMF,DISP=OLD
//*
//CAIVMFS DD DSN=&VMF,DISP=SHR,
// DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIBVMF DD DSN=&BKUPVMF,DISP=OLD
//*
//CAIMVMF DD DSN=&MERGE,DISP=OLD
//*
//CAIIDCK DD DSN=&IDCK,DISP=SHR
//*
//SORTWK01 DD UNIT=&WORK,SPACE=&SPCVMF
//*
//SYSPRINT DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT DD SYSOUT=&A
//*
//SYSUDUMP DD SYSOUT=&A
//*
//*************************************************
```

Starter JCL

```
//VMFMRGE JOB
//VMFMRGE EXEC CATVMFMR
```

CATVMFMR Output

The ++MERGE parameter produces the Merge Volume Master report (TLMS052).

```
YOUR TLMS COMPANY NAME VOLUME MASTER FILE UTILITY FOR ++MERGE                    TLMS052      PAGE       1
CA TLMS NN.n yymmTLrrr                                                            mm/dd/yyyy  hh.mm.ss

                                                CONTROLDATA
                                        ----------------------------------------
                                        CHAINING RECORDS RESERVED.................100
                                        MESSAGE RECORDS RESERVED..................50
                                        CHAINING THRESHOLD COUNT..................70
                                        VOLUMES INITIALIZED AS.................SCRATCH
                                        COMPANY.........TLMS TEST SYSTEM

                                        TOTAL INPUT RECORDS READ..................834
                                        TOTAL CONTROL RECORDS WRITTEN.............1
                                        TOTAL SEGMENT RECORDS WRITTEN.............1
                                        TOTAL BASE RECORDS WRITTEN................461
                                        TOTAL MVL RECORDS WRITTEN.................0
                                        TOTAL MDS RECORDS WRITTEN.................12
                                        TOTAL AUXILIARY RECORDS WRITTEN...........89
                                        TOTAL MESSAGE RECORDS WRITTEN.............50
                                        TOTAL RECORDS WRITTEN.....................614
                                        TOTAL ORPHANED MVL RECORDS................0
                                        TOTAL ORPHANED MDS RECORDS................24
                                        TOTAL ORPHANED VSN RECORDS................0
                                        TOTAL WARNING MESSAGES....................24
```

Note: For information on the fields in this report, see the section CATVMFB Output (see page 218).

CATVMFRE - Reorganize the Volume Master File

In the event that the Volume Master File is corrupted or destroyed, it may be necessary to reorganize it from the most recent backup copy. This process rebuilds all of the chains and attempts to repair invalidly chained multivolume data sets. Run the CAI.CTAPPROC (CATVCVS) PROC to obtain the volumes with chaining errors and then make any necessary repairs before the reorganization. The technique used for ++REORG is different from ++RESTORE in not using TLMSIDCK to define the VMF. No changes can be made from the backup versions.

The VMF Index file must be re-created immediately after the VMF has been reorganized. See "CATVMFXI - Initialize or Re-create the VMF Index File" in this chapter.

CATVMFRE Procedure JCL

The following procedure is used to reorganize the Volume Master File.

```
//*************************************************************
//*    **** PROCNAME=CATVMFRE ****      *
//*************************************************************
//** PROCEDURE TO REORGANIZE THE CA TLMS VOLUME MASTER FILE   *
//*************************************************************
//CATVMFRE PROC A='*',
//    BUFNO='80',
//    BKUPVMF='CAI.TLMS.BKUPVMF',
//    LOAD='CAI.CTAPLINK',
//    OPTS='CAI.CTAPOPTN(TLMSIPO)',
//    SORTLIB='SYS1.SORTLIB',
//    VMF='CAI.TLMS.VMF',
//    SPCVMF='(CYL,(5,5))',
//    WORK='SYSDA'
//*
//*************************************************************
//**   EXECUTE VOLUME MASTER FILE UTILITY     *
//*************************************************************
//REORGVMF EXEC PGM=TLMSVMFU,PARM='++REORG'
//*
//STEPLIB DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD DSN=&OPTS,DISP=SHR
//*
//SORTLIB DD DSN=&SORTLIB,DISP=SHR
//*
//CAIVMF DD DSN=&VMF,DISP=SHR
//*
//CAIVMFS DD DSN=&VMF,DISP=SHR,
//      DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIBVMF DD DSN=&BKUPVMF,DISP=OLD
//*
//SORTWK01 DD UNIT=&WORK,SPACE=&SPCVMF
//*
//SYSPRINT DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT DD SYSOUT=&A
//*
//SYSUDUMP DD SYSOUT=&A
//*
//*************************************************************
```

Starter JCL

```
//VMFREORG JOB
//VMFREORG EXEC CATVMFRE
//
```

CATVMFRE Output

The ++REORG parameter produces the REORG Volume Master File report (TLMS047).

```
YOUR TLMS NN.n COMPANY NAME            VOLUME MASTER FILE UTILITY FOR ++REORG              TLMS047    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                     mm/dd/yyyy  hh.mm.ss
                                                TOTAL INPUT RECORDS READ..................616
                                                TOTAL CONTROL RECORDS WRITTEN.............1
                                                TOTAL SEGMENT RECORDS WRITTEN.............1
                                                TOTAL BASE RECORDS WRITTEN................461
                                                TOTAL MVL RECORDS WRITTEN.................0
                                                TOTAL MDS RECORDS WRITTEN.................12
                                                TOTAL AUXILIARY RECORDS WRITTEN...........89
                                                TOTAL MESSAGE RECORDS WRITTEN.............50
                                                TOTAL RECORDS WRITTEN.....................614
                                                TOTAL ORPHANED MVL RECORDS................0
                                                TOTAL ORPHANED MDS RECORDS................0
                                                TOTAL ORPHANED VSN RECORDS................0
                                                TOTAL WARNING MESSAGES....................0
```

Note: For information on the fields in this report, see the section CATVMFB Output (see page 218).

CATVMFRL - Reload the Volume Master File

This process copies a VMF backup to the VMF. It does not reorganize the VMF data. It does not delete orphan records. It does not correct or report chaining errors.

Reload should be used when you need an exact copy of the VMF from the time the backup was taken.

CATVMFRL Procedure JCL

The following procedure is used to reload the Volume Master File.

```
//*************************************************************
//*          **** PROCNAME=CATVMFRL ****                      *
//*************************************************************
//**     PROCEDURE TO RELOAD CA TLMS VOLUME MASTER FILE     *
//*************************************************************
//*
//CATVMFRL PROC A='*',
//             BKUPVMF='CAI.TLMS.BKUPVMF',
//             BUFNO='80',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             VMF='CAI.TLMS.VMF'
//*
//*************************************************************
//**        EXECUTE VOLUME MASTER FILE UTILITY               *
//*************************************************************
//RELOAD     EXEC PGM=TLMSVMFU,PARM='++RELOAD'
//*
//STEPLIB    DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS   DD DSN=&OPTS,DISP=SHR
//*
//CAIVMFS    DD DSN=&VMF,DISP=OLD,
//              DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIBVMF    DD DSN=&BKUPVMF,DISP=OLD
//*
//SYSPRINT   DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT     DD SYSOUT=&A
//*
//SYSUDUMP   DD SYSOUT=&A
//*
//*************************************************************
```

Starter JCL

```
VMFRELOD JOB
//VMFRELOD EXEC CATVMFRL
```

CATVMFRL Output

The ++RELOAD parameter produces the Reload Volume Master File report (TLMS050).

```
YOUR TLMS NN.n COMPANY NAME            VOLUME MASTER FILE UTILITY FOR ++RELOAD            TLMS050    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                    mm/dd/yyyy  hh.mm.ss
                                                TOTAL INPUT RECORDS READ.................615
                                                TOTAL CONTROL RECORDS WRITTEN..............1
                                                TOTAL SEGMENT RECORDS WRITTEN..............1
                                                TOTAL BASE RECORDS WRITTEN.................461
                                                TOTAL MVL RECORDS WRITTEN..................0
                                                TOTAL MDS RECORDS WRITTEN..................12
                                                TOTAL AUXILIARY RECORDS WRITTEN............90
                                                TOTAL MESSAGE RECORDS WRITTEN..............50
                                                TOTAL RECORDS WRITTEN......................615
                                                TOTAL ORPHANED MVL RECORDS.................0
                                                TOTAL ORPHANED MDS RECORDS.................0
                                                TOTAL ORPHANED VSN RECORDS.................0
                                                TOTAL WARNING MESSAGES.....................0
```

Note: For information on the fields in this report, see the section CATVMFB Output (see page 218).

CATVMFRS - Restore the Volume Master File

If the Volume Master File is corrupted or destroyed, you must restore it from the most recent backup copy. The VMF can be initialized at the same time it is restored.

If the most recent updates to the VMF are not contained in the VMF backup copy, you must use the CATVMFRV procedure.

This process rebuilds all chains and attempts to repair invalidly chained multi-volume data sets.

The restore function releases auxiliary records no longer being used. The TLMS048 and TLMS049 reports may show different totals.

The VMF Index file must be re-created immediately after the VMF has been restored. See CATVMFXI - Initialize or Re-create the VMF Index File in this chapter.

CATVMFRS Procedure JCL

The following procedure is used to restore the Volume Master File.

```
//*********************************************************
//*         **** PROCNAME=CATVMFRS ****                   *
//*********************************************************
//* PROCEDURE TO RESTORE THE CA TLMS VOLUME MASTER FILE   *
//*********************************************************
//CATVMFRS PROC A='*',
//             BKUPVMF='CAI.TLMS.BKUPVMF',
//             BUFNO='80',
//             IDCK='CAI.CTAPOPTN(TLMSIDCK)',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             SORTLIB='SYS1.SORTLIB',
//             SPCVMF='(CYL,(5,5))',
//             WORK='SYSDA',
//             VMF='CAI.TLMS.VMF'
//*
//*********************************************************
//**                EXECUTE VOLUME MASTER FILE UTILITY    *
//*********************************************************
//RESTOVMF    EXEC PGM=TLMSVMFU,PARM='++RESTORE'
//*
//STEPLIB     DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS    DD DSN=&OPTS,DISP=SHR
//*
//SORTLIB     DD DSN=&SORTLIB,DISP=SHR
//*
//CAIVMF      DD DSN=&VMF,DISP=SHR
//*
//CAIVMFS     DD DSN=&VMF,DISP=SHR,
//               DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIBVMF     DD DSN=&BKUPVMF,DISP=OLD
//*
//CAIIDCK     DD DSN=&IDCK,DISP=SHR
//*
//SORTWK01    DD UNIT=&WORK,SPACE=&SPCVMF
//*
//SYSPRINT    DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT      DD SYSOUT=&A
//*
//SYSUDUMP    DD SYSOUT=&A
//*
//*********************************************************
```

Starter JCL

```
//VMFREST JOB
//VMFREST EXEC CATVMFRS
//
```

CATVMFRS Output

The ++RESTORE command produces the Restore Volume Master File report (TLMS049).

```
YOUR TLMS NN.n COMPANY NAME          R E C O V E R V O L U M E M A S T E R *FROM* B A C K U P       TLMS030    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                 dd mmm yyyy hh.mm.ss
                                     MASTER DUMP DATE AND TIME.....................mm/dd/yyyy 05.24.15
                                     BACKUP TRANSACTIONS STARTING DATE AND TIME....mm/dd/yyyy 01.14.00
                                     BACKUP TRANSACTIONS ENDING DATE AND TIME......mm/dd/yyyy 11.16.52

                                     NUMBER OF TRANSACTIONS APPLIED TO RECOVERY FILE.... 17,369
                                                                E N D     O F       J O B
```

```
YOUR TLMS NN.n COMPANY NAME          VOLUME MASTER FILE UTILITY FOR ++RESTORE            TLMS049    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                   mm/dd/yyyy  hh.mm.ss
                                                      CONTROLDATA
                                                ------------------------------------------------
                                                CHAINING RECORDS RESERVED.................100
                                                MESSAGE RECORDS RESERVED..................50
                                                CHAINING THRESHOLD COUNT..................70
                                                VOLUMES INITIALIZED AS.................SCRATCH
                                                COMPANY.........TLMS TEST SYSTEM

                                                TOTAL INPUT RECORDS READ..................614
                                                TOTAL CONTROL RECORDS WRITTEN.............1
                                                TOTAL SEGMENT RECORDS WRITTEN.............1
                                                TOTAL BASE RECORDS WRITTEN................461
                                                TOTAL MVL RECORDS WRITTEN.................0
                                                TOTAL MDS RECORDS WRITTEN.................12
                                                TOTAL AUXILIARY RECORDS WRITTEN...........89
                                                TOTAL MESSAGE RECORDS WRITTEN.............50
                                                TOTAL RECORDS WRITTEN.....................614
                                                TOTAL ORPHANED MVL RECORDS................0
                                                TOTAL ORPHANED MDS RECORDS................0
                                                TOTAL ORPHANED VSN RECORDS................0
                                                TOTAL WARNING MESSAGES....................0
```

Note: For information on the fields in this report, see the section CATVMFB Output (see page 218).

Expanding or Decreasing the Volume Master File

The VMF initialization program TLMSVMFU is also used to

- Add new ranges of volume serial numbers
- Delete existing ranges of volume serial numbers
- Expand/decrease the multivolume/multi-data set record area
- Add or delete the auxiliary message area

Important! Should an overflow warning message occur indicating that the VMF is near its capacity, you must delete entries or expand the file size to prevent data loss.

Process

To update, expand or decrease the VMF, use the following steps:

1. Quiesce all system tape activity.
2. Back up the current Volume Master File.

   Note: For more information, see CATVMFB - Back Up the Volume Master File (see page 216).
3. Locate the VMF initialization commands and modify them to reflect all additions and deletions.

   Note: For more information, see CATVMFI - Initialize the Volume Master File (see page 220).
4. Use the backup tape created in Step 2 above to restore the VMF.

   Note: For more information, see CATVMFRS - Restore the Volume Master File (see page 237).
5. Execute PROC CATVMFXI to rebuild VMF index.
6. Allow tape activity to resume.

To restore the VMF while CA TLMS is active, use a different data set name. You may also need to apply the tape transactions that occurred since the backup was created.

Note: For more information, see CATVMFRV - Recover the Volume Master File (see page 241).

CATVMFRV - Recover the Volume Master File

If the Volume Master File must be restored and there has been tape activity since the last VMF backup, you must reprocess the VMF update transactions from a backup of the alternate log file (ALOG) or the SMF data sets, whichever is applicable. (These programs cannot be executed against a live ALOG file or SMF data sets.)

Process

The recovery process uses the Volume Master File recovery program TLMSRECV. After restoring the VMF, execute the TLMSRECV program, supplying as input all of the log data sets that were created since the last backup of the VMF. After successful completion of the VMF recovery, it will be necessary to re-create the VMF index file (see "CATVMFXI - Initialize or Re-create the VMF Index File" on CATVMFXI - Initialize or Recreate the VMF Index File in this guide).

Note: TLMSRECV will only recover the ALOG transactions that are newer than the date/time stamp from the VMF Backup (DD CAIBVMF).

CATVMFRV Procedure JCL

The following procedure both restores and recovers the VMF to a newly initialized VMF file.

```
//*************************************************************
//*                    **** PROCNAME=CATVMFRV ****            *
//*************************************************************
//** RECOVER VOLUME MASTER FILE TO POINT OF FAILURE           *
//*************************************************************
//** CA TLMS VMF MUST BE REINITED (CATVMFI) BEFORE THIS STEP. *
//*************************************************************
//CATVMFRV PROC A='*',
//       BKUPVMF='CAI.TLMS.BKUPVMF',
//       BUFNO='80',
//       IDCK='CAI.CTAPOPTN(TLMSIDCK)',
//       LOAD='CAI.CTAPLINK',
//       OPTS='CAI.CTAPOPTN(TLMSIPO)',
//       SPCVMF='(CYL,(5,5))',
//       SORTLIB='SYS1.SORTLIB',
//       BKUPALOG='CAI.TLMS.BKUPALOG',
//       VMF='CAI.TLMS.VMF',
//       WORK='SYSDA'
//*
//*
```

```
//*******************************************************
//**                EXECUTE VOLUME MASTER FILE UTILITY       *
//*******************************************************
//RESTOVMF    EXEC PGM=TLMSVMFU,PARM='++RESTORE'
//*
//STEPLIB     DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS    DD DSN=&OPTS,DISP=SHR
//*
//SORTLIB     DD DSN=&SORTLIB,DISP=SHR
//*
//CAIVMF      DD DSN=&VMF,DISP=SHR
//*
//CAIVMFS     DD DSN=&VMF,DISP=SHR,
//               DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//CAIBVMF     DD DSN=&BKUPVMF,DISP=OLD
//*
//CAIIDCK     DD DSN=&IDCK,DISP=SHR
//*
//SORTWK01    DD UNIT=&WORK,SPACE=&SPCVMF
//*
//SYSPRINT    DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT      DD SYSOUT=&A
//*
//SYSUDUMP    DD SYSOUT=&A
//*
//*
```

```
//*****************************************************
//** REBUILD VOLUME MASTER TO POINT OF FAILURE       *
//** (USES BKUPALOG OR SMF)                          *
//*****************************************************
//RECOVVMF   EXEC PGM=TLMSRECV,
//               COND=(4,LT)
//*
//STEPLIB    DD DSN=&LOAD,DISP=SHR
//*
//SORTLIB    DD DSN=&SORTLIB,DISP=SHR
//*
//TLMSOPTS   DD DSN=&OPTS,DISP=SHR
//*
//CAIVMF     DD DSN=&VMF,DISP=SHR
//*
//CAIVMFS    DD DSN=&VMF,DISP=SHR
//*
//CAIBVMF    DD DSN=&BKUPVMF,DISP=OLD
//*
//VMFTRANS DD DSN=&BKUPALOG,DISP=OLD    BKUP ALOG OR SMF
//*
//SORTWK01 DD UNIT=&WORK,SPACE=&SPCVMF
//*
//SYSPRINT DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT DD SYSOUT=&A
//*
//SYSUDUMP DD SYSOUT=&A
//*
//SYSIN DD DUMMY
//*
//*************************************************************
```

Starter JCL

```
//VMFRECV JOB
//VMFRECV EXEC CATVMFRV
//
```

CATVMFRV Output

This process produces the restore report (TLMS049) and a recovery report (TLMS030).

```
YOUR TLMS NN.n COMPANY NAME            VOLUME MASTER FILE UTILITY FOR ++RESTORE            TLMS049    PAGE      1
CA TLMS NN.n yymmTLrrr                                                                     mm/dd/yyyy  hh.mm.ss
                                                    CONTROLDATA
                                          ---------------------------------------------
                                          CHAINING RECORDS RESERVED.................100
                                          MESSAGE RECORDS RESERVED...................50
                                          CHAINING THRESHOLD COUNT...................70
                                          VOLUMES INITIALIZED AS.................SCRATCH
                                          COMPANY.........TLMS TEST SYSTEM

                                          TOTAL INPUT RECORDS READ..................614
                                          TOTAL CONTROL RECORDS WRITTEN.............1
                                          TOTAL SEGMENT RECORDS WRITTEN.............1
                                          TOTAL BASE RECORDS WRITTEN................461
                                          TOTAL MVL RECORDS WRITTEN.................0
                                          TOTAL MDS RECORDS WRITTEN.................12
                                          TOTAL AUXILIARY RECORDS WRITTEN...........89
                                          TOTAL MESSAGE RECORDS WRITTEN.............50
                                          TOTAL RECORDS WRITTEN.....................614
                                          TOTAL ORPHANED MVL RECORDS................0
                                          TOTAL ORPHANED MDS RECORDS................0
                                          TOTAL ORPHANED VSN RECORDS................0
                                          TOTAL WARNING MESSAGES....................0
```

Note: For information on the fields in this report, see CATVMFB Output (see page 218).

```
your company name              R E C O V E R V O L U M E M A S T E R *FROM* B A C K U P       TLMS030    PAGE      1
CA TLMS NN.n yymmTLrrr                                                dd mmm yyyy hh.mm.ss
                               MASTER DUMP DATE AND TIME.....................mm/dd/yyyy 05.24.15
                               BACKUP TRANSACTIONS STARTING DATE AND TIME....mm/dd/yyyy 01.14.00
                               BACKUP TRANSACTIONS ENDING DATE AND TIME......mm/dd/yyyy 11.16.52

                               NUMBER OF TRANSACTIONS APPLIED TO RECOVERY FILE.... 17,369
                                                E N D     O F       J O B
```

CATVMFSC - Set Volume Master File Control Information

This process changes some of the values for control information in an active VMF. The Company Name, CPU ID, and Warning Limit may be changed for the active VMF without having to reinitialize the VMF.

CATVMFSC Procedure JCL

The following procedure is used to modify control data in the active VMF.

```
//*******************************************************
//*          **** PROCNAME=CATVMFSC ****                *
//*******************************************************
//**        PROCEDURE TO SET VALUES OF CONTROL FIELDS *
//*******************************************************
//*
//CATVMFSC PROC A='*',
//             IDCK='CAI.CTAPOPTN(TLMSIDCK)',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             VMF='CAI.TLMS.VMF'
//*
//*******************************************************
//**          EXECUTE VOLUME MASTER FILE UTILITY      *
//*******************************************************
//SETCTL     EXEC PGM=TLMSVMFU,PARM='++SETCTL '
//*
//STEPLIB    DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS   DD DSN=&OPTS,DISP=SHR
//*
//CAIVMF     DD DSN=&VMF,DISP=OLD
//*
//CAIIDCK    DD DSN=&IDCK,DISP=SHR
//*
//SYSPRINT   DD DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSUDUMP   DD SYSOUT=&A
//*
//*******************************************************
```

Starter JCL

```
//VMFSC JOB
//VMGSC EXEC CATVMFSC
```

CATVMFSC Output

The ++SETCTL parameter produces the Set Control Volume Master File report (TLMS051).

```
YOUR TLMS NN.n COMPANY NAME   VOLUME MASTER FILE UTILITY FOR ++SETCTL            TLMS051    PAGE       1
CA TLMS NN.n yymmTLrrr                                                          mm/dd/yyyy  hh.mm.ss
                                          CONTROLDATA
                                  ---------------------------------------------
                                  CHAINING RECORDS RESERVED.................100
                                  MESSAGE RECORDS RESERVED...................50
                                  CHAINING THRESHOLD COUNT..................393
                                  VOLUMES INITIALIZED AS.................SCRATCH
                                  COMPANY.........TLMS TEST SYSTEM
```

CATVMFV - Verify the Volume Master File

Use the CATVMFV procedure to verify a VMF or backup of a VMF and produce the TLMS064 report.

TLMSVMFV is executed to verify the VMF using the same technique used by TLMSVMFU for the VMF utilities. The technique allows TLMSVMFV to verify VMF record chaining without "running the chains". This prevents I/O errors from bad chains and allows TLMSVMFV to continue verification of a chain after an error is detected. TLMSVMFV also finds orphaned records. Orphaned records are records which cannot be found by TLMS batch processing that starts with the beginning of a tape chain.

The scope of the verification and reporting is controlled by the following CATVMFV parameters:

DELAY

This parameter instructs TLMSVMFV to delay the verification of tape chains for the specified number of days. This prevents reporting of false errors for tapes which are being created while CATVMFV is running.

LEVEL

This parameter specifies the level of detail TLMSVMFV is to report. The LEVEL parameter can greatly reduce the size of the TLMS064 report. By default TLMSVMFV uses DELAY=3 and LEVEL=1. The report levels are:

- Level 1 is the default report level.
- Level 5 and above print orphans records. These are records which are not connected to valid TLMS chains. These records are lost during a VMF RESTORE.
- Level 9 is the most detailed report. Level 9 overrides the DELAY PARM. Level 9 can only be run against a VMF that has no activity.

Process

Modify the CATVMFV parameter for SORT to values similar to those in CATVMFU. These vary with the size of the VMF, for example:

```
SORTLIB='SYS1.SORTLIB'
SPCVMF='(CYL,(5,5))'
WORK='SYSALLDA'
```

Specify the DELAY and LEVEL parameters via CATVMFV PRM=. For example:

```
CATVMFV  PRM='DELAY=6   LEVEL=5'
```

DELAY

Specifies the number of days TLMSVMFV waits after the creation date before any given verification.

Range: 0 to 9999999 days

LEVEL

Specifies the level of detail for verification and reporting. The lower the value of LEVEL the less detail produced.

Valid values: 1, 2, 5, and 9

PRM keywords can be specified in any order and are separated by commas or blanks.

CATVMFV Procedure JCL

Use the following procedure to verify the Volume Master File:

```
//*********************************************************************
//*            ****  PROCNAME=CATVMFV  ****                           *
//*********************************************************************
//**      PROCEDURE TO VERIFY THE CA TLMS VOLUME MASTER FILE          *
//*********************************************************************
//CATVMFV  PROC A='*',
//             BUFNO='80',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             PRM='',
//             SORTLIB='SYS1.SORTLIB',
//             SPCVMF='(CYL,(5,5))',
//             WORK='SYSALLDA',
//             VMF='CAI.TLMS.VMF'
//*
//*********************************************************************
//**           VERIFY VOLUME MASTER FILE                             **
//*********************************************************************
//VERIFY   EXEC PGM=TLMSVMFV,PRM='&PRM'
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//CAIVMFS  DD  DSN=&VMF,DISP=SHR,
//             DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR
//*
//SORTLIB  DD  DSN=&SORTLIB,DISP=SHR
//*
//SORTWK01 DD  UNIT=&WORK,SPACE=&SPCVMF
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSOUT   DD  SYSOUT=&A
//*
//SYSUDUMP DD  SYSOUT=&A
//*
//*********************************************************************
```

Starter JCL

```
//VMFVER JOB
//VERIFY EXEC CATVMFV,PRM='DELAY=3,LEVEL=1'
//
```

Output

The CATVMFV utility produces report TLMS064. This report has two sections:

VOLUME MASTER FILE VERIFICATION LOG

This section is written only if problems are found with the VMF.

SUMMARY OF VOLUME MASTER FILE VERIFICATION

This section provides a count of the various elements found in the VMF.

```
YOUR COMPANY NAME                    VOLUME MASTER FILE VERIFICATION LOG     SCAN LEVEL(1)          TLMS064        PAGE   1

CA TLMS       NN.n  mmyyTLrrr                                                      mm/dd/yy    hh.mm.ss


 CHAIN-ID  CHNVOL TYPE VOLSER    VSQ.    FSQ.     ----------------------------------------------------------------

           070001   B  070001       4       1     CAT6420W -  UNCHAINED VOLUME HAS FIELD(S) SHOWING CHAINING

 00007101  000003   B  000001       1       1     CAT6437W -  VSN OF CHAIN BASE MUST EQUAL CHNVOL

 00007101  000003   B  FRED01       2             CAT6418E -  VOLSER MISSING FROM MULTI-VOLUME CHAIN

                     JANE02       3             CAT6418E -  VOLSER MISSING FROM MULTI-VOLUME CHAIN

 00001101  001527   B  001528       2             CAT6418E -  VOLSER MISSING FROM MULTI-VOLUME CHAIN

                     001529       3             CAT6418E -  VOLSER MISSING FROM MULTI-VOLUME CHAIN

                     001530       4             CAT6418E -  VOLSER MISSING FROM MULTI-VOLUME CHAIN

                     001531       5             CAT6418E -  VOLSER MISSING FROM MULTI-VOLUME CHAIN

 00001C01  001598   B  001599       1       1     CAT6437W -  VSN OF CHAIN BASE MUST EQUAL CHNVOL

 00001C01  001598   B  001599       1       1     CAT6439W -  CHAIN BASE HAS NO DATA SETS OR VOLUMES CHAINED

 00001D01  090001   D  090001       1       6     CAT6445E -  LAST MDS PTR IN BASE DOES NOT POINT TO THIS MDS

 00001D01  090001   D  090001       1       8     CAT6443E -  PREV MDS DOES NOT POINT TO THIS MDS

 00000601  TDI028   D  TDI028       1      40     CAT6445E -  LAST MDS PTR IN BASE DOES NOT POINT TO THIS MDS

 00000601  TDI028   D  TDI028       1      42     CAT6443E -  PREV MDS DOES NOT POINT TO THIS MDS
```

```
YOUR COMPANY NAME                    SUMMARY OF VOLUME MASTER FILE VERIFICATION              TLMS064        PAGE   2

CA TLMS       NN.n  mmyyTLrrr                                                              09/30/2010     09.02.14


                                     TOTAL VMF RECORDS READ .................2,602

                                     TOTAL CONTROL RECORDS ......................1

                                     TOTAL SEGMENT RECORDS ......................2

                                     TOTAL BASE RECORDS .....................1,534

                                     TOTAL MVL RECORDS ..........................1

                                     TOTAL MDS RECORDS ........................488

                                     TOTAL AUXILIARY RECORDS ..................526

                                     TOTAL MESSAGE RECORDS .....................50

                                     TOTAL PADDING RECORDS .....................15

                                     TOTAL AUXILIARY RECORDS ON REUSE ............5

                                     TOTAL SKIP VOLUMES .........................1

                                     TOTAL ORPHANED MVL RECORDS..................0

                                     TOTAL ORPHANED MDS RECORDS..................0

                                     TOTAL ORPHANED VSN RECORDS..................4

                                     TOTAL WARNINGS..............................4

                                     TOTAL ERRORS ..............................10
```

CATVMFX - Build a New VMF with EXTEND

How to Build a VMF with EXTEND

Use the CATVMFX utility to build dynamically a new VMF while tape processing continues across the shared VMF environment. The CATVMFX procedure controls the dynamic VMF EXTEND process. This process consists of a series of commands that are issued to all TLMSs sharing the VMF.

Plan for Executing CATVMFX Extend

Planning is an important part of running the CATVMFX utility. The CATVMFX utility creates new volume base records in the VMF. Every TLMS sharing the VMF must be at the correct maintenance level to perform an EXTEND. CATVMFX checks the maintenance level of every TLMS it finds by reading the VMF PRINTCPU field in the control record. If any of the TLMSs are not at the correct maintenance level, a STATE of "Old Code" is issued in the status message. The new VMF and the alias must belong to the same z/OS user catalog, for example, their high level qualifiers must match.

The new VMF must be large enough to contain all active records currently in the old VMF. The new VMF can reside on the same volume or a different volume than the original VMF. Catalog the new VMF to the same user catalog as the alias. You can execute CATVMFX as a z/OS started task or a batch job.

During the extend process, all TLMS tasks perform synchronous updates to both VMFs while records from the old VMF are copied to the new VMF. Once all active records are copied, a z/OS Catalog alias points to the new VMF. All batch and online tape activity references the new VMF through this z/OS Catalog alias. After the EXTEND process completes, all CA TLMS tasks drop the old VMF, ending the process.

Since both the old and new VMFs are updated synchronously during the extend process, you do not need to stop batch and online tape activity. All jobs referencing either the old or the new VMF can run to completion safely.

CATVMFX Modes

CATVMFX runs in three modes to perform different functions, EXTEND, TEST, and RESET. To verify the shared VMF environment before you perform a switch, run CATVMFX in TEST mode. The utility is designed to test as many environmental details as possible before proceeding with a VMF switch, reducing the likelihood of a problem.

Select a mode by specifying one of the following values for the 'DO=' parameter in the CATVMFX PROC.

EXTEND

Instructs CATVMFX to initiate the dynamic VMF EXTEND process and terminate once all TLMSs have switched VMFs. This process performs the actual EXTEND, and all TLMSs use the new VMF when completed.

TEST

Instructs CATVMFX to initiate the dynamic VMF EXTEND process, but skips the switch and recatalog steps. This process only tests the EXTEND process, and all TLMSs continue using the old VMF when completed.

RESET

Instructs CATVMFX to clean up the VMF control record. This process clears all residual data that is left behind from an aborted EXTEND/TEST. (CATVMFX uses the VMF control record to issue commands to all TLMSs). Due to the nature and impact of this utility, contact CA Support for any error conditions that occur.

Note: The CATVMFX EXTEND process does not perform any VMF reorganization functions. Perform the VMF reorganizations to remove orphan chains, clear the reuse chain, reorganize AUX records, and reassign chain-ids.

We recommend that you run the batch version of VMF Extend copy (HLQ.CTAPJCL(TLMJVMW3)) before running an EXTEND. This version validates that all records in the old VMF can be copied to the new VMF. CATVMFX stops after the first record in error, but the batch version logs all the errors in CAIMSG.

CATVMFX EXTEND Requirements

Before performing the dynamic extend, ensure that the following conditions are met:

- The new VMF is initialized with CATVMFI.
- The new VMF and the alias belong to the same user catalog, for example, their high level qualifiers must match.
- The last VMF backup was no longer than three hours before the dynamic extend.
- The new VMF is large enough to contain all active records in the old VMF.
- The new VMF has enough AUX records to contain those records that are used in the old VMF.
- The new VMF contains the VSNs of all the active volumes in the old VMF.
- The new VMF is cataloged to the same user catalog as the old VMF.
- XUPD user exits, that open the VMF for record processing, must process CLOSE calls from TLMSUPDT. CATVMFX requires a successful deallocation of the old VMF to complete. See the sample in HLO.CTAPSAMP(TLMSXUPD). If you use a custom TLMSXUPD exit, contact CA Support to review your exit.

CATVMFX Restrictions

If you plan to remove volume ranges from the old VMF, observe the following restrictions:

- The volser range must be in scratch status and all chaining records must be broken and cleared.
- TLMSXUPD user exits that open the VMF for record processing, process CLOSE calls from TLMSUPDT. CATVMFX requires the successful deallocation of the old VMF to complete. If you use a custom TLMSXUPD exit, contact CA Support to review your exit.
- Do not run PROC CATTRS during the VMF extend.
- Do not run batch jobs because CATVMFX has to wait for the jobs to complete before dropping the old VMF.
- Ensure that no volume ranges are defined in a range statement in the CTOSCRnn member of CTAPOPTN, if you use the scratch subpooling feature to remove volume ranges.

CATVMFX EXTEND Process

To perform an EXTEND or a TEST, run CATVMFX as a started task or batch job. Edit the parameters in the PARMS data set. Run CATVMFX as a started task overriding the DO parameter with the mode. CATVMFX issues commands to all instances of TLMS that are sharing the VMF and writes the commands in the VMF control record. Each instance of TLMS responds that the command completed or failed. CATVMFX waits for all TLMs to complete processing the command before issuing the next command. TLMSs post their completion status to the VMF control record. Once all steps are complete, CATVMFX terminates.

CATVMFX performs the following steps during EXTEND:

1. Prompts you to verify the parameters for the EXTEND.
2. Issues an attention command to all TLMSs. This command instructs all TLMSs to start checking the VMF control record every 30 seconds instead of every 3 minutes.
3. Issues a test access command to all TLMSs. This command instructs all TLMSs to verify update access to the new VMF and ability to create a VMF alias.
4. Issues a drop allocation for DDs CAIVMFS and TAPEDB.
5. Issues an add VMF command to all TLMSs. This command instructs all TLMSs to add a secondary VMF and write duplex records during tape processing.
6. Copies all active records from the old VMF to the new VMF. Changes to active records during this step (resulting from tape activity) are dynamically updated through duplex writes. The primary and secondary VMFs remain in sync.
7. Issues a switch command to all TLMSs. This command instructs all TLMSs to use the new VMF as their primary VMF, and write duplex records to the old VMF.
8. Issues a recatalog command to all TLMSs. This command instructs all TLMSs to point the VMF alias on their system to the new VMF. New batch jobs and online sessions now reference the new VMF through the alias.
9. Issues a drop VMF command to all TLMSs. This command instructs all TLMSs to wait for all jobs using the old VMF to terminate. Then it drops the secondary VMF (stops writing duplex records).
10. Poisons the old VMF to prevent new jobs from using it.
11. Issues a return-to-normal command to all TLMSs. This command instructs all TLMSs to resume checking the VMF control record every 3 minutes.

CATVMFX TEST Process

CATVMFX performs the following steps during the TEST:

1. Performs steps 1 through 6 of the CATVMFX EXTEND process (see page 255).
2. Skips steps 7 and 8 of the CATVMFX EXTEND process. The secondary VMF is still the new VMF (VMF2).
3. Issues a drop VMF command to all TLMSs. This command instructs all TLMSs to drop the secondary VMF, that is, stop writing duplex records.
4. Does not poison the secondary VMF in the TEST mode letting you check the new VMF for accuracy.
5. Issues a return-to-normal command to all TLMSs. This command instructs all TLMSs to resume checking the VMF control record every 3 minutes.

Procedure JCL

```
//*****************************************************************
//*              ****  PROCNAME=CATVMFX ****                      *
//*****************************************************************
//**     PROCEDURE TO DYNAMICALLY EXTEND THE VMF                  *
//*****************************************************************
//*
//CATVMFX  PROC A='*',
//             LOAD='CAI.CTAPLINK',
//             PARMS='CAI.CTAPOPTN(CTOVMFX)',
//             DO='TEST'
//*
//*****************************************************************
//**           EXECUTE DYNAMIC VMF EXTEND                         *
//*****************************************************************
//EXTEND   EXEC PGM=TLMSVMFX,PARM='&DO'
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//CAIMSG   DD  SYSOUT=&A
//*
//CAIPARM  DD  DSN=&PARMS,DISP=SHR
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A
//*
//SYSUDUMP DD  SYSOUT=&A
//*
```

Starter JCL

```
//VMFEXTND  JOB
//VMFEXTND  EXEC CATVMFX
```

Start Procedures

1. Start CATVMFX in EXTEND mode using the command:

   ```
   S CATVMFX,DO=EXTEND
   ```

2. Start CATVMFX in TEST mode using the command:

   ```
   S CATVMFX,DO=TEST
   ```

3. Start CATVMFX in RESET mode using the command:

   ```
   S CATVMFX,DO=RESET
   ```

CATVMFX Output

The CATVMFX outputs are WTOs written to the z/OS console and events that are logged to the CAIMSG DD statement.

CATVMFX Verification Display

```
CAT6550I Parameters Specified in CAIPARM
CAT6551I MODE         EXTEND
CAT6551I VMF1         CAI.TLMS.VMF.A
CAT6551I VMF2         CAI.TLMS.VMF.B
CAT6551I ALIAS        CAI.TLMS.VMF
CAT6551I COPYWAIT     00000200
CAT6551I COPYBLKS     00000010
CAT6551I EXCLUDE      CPUH
CAT6501I VMF Extend process initiated, 01 LPARs detected
CAT6553I STEP: VMF Extend Start up
CAT6554I LPAR  CODE  VMF  STATE      STATUS
CAT6555I CPU1   00                   Inactive
CAT6555I CPU7   00                   Inactive
CAT6555I CPU2   00                   Active
CAT6555I CPUO   40        Old Code   Inactive
CAT6555I CPUH   00                   Excluded
18 CAT6592P Verify parameters and enter Y to confirm the VMF extend
```

CATVMFX CAIMSG Log

```
TLMSVMFX OPEN CAIPARM
TLMSVMFX CLOSE CAIPARM
CAT6550I Parameters Specified in CAIPARM
CAT6551I MODE         EXTEND
CAT6551I VMF1         ASM.TLMSDV.VMF.A
CAT6551I VMF2         ASM.TLMSDV.VMF.Z
CAT6551I ALIAS        ASM.TLMSDV.VMF.A
CAT6551I COPYWAIT     00000200
CAT6551I COPYBLKS     00000010
CAT6551I EXCLUDE      BOBS
TLMSVMFX ALLOCATE VMF1
TLMSVMFX OUTPUT FROM IDCAMS
 LISTCAT ENT(ASM.TLMSDV.VMF.A                              )  ALL
NONVSAM ------- ASM.TLMSDV.VMF.A
     IN-CAT --- ICF.VMVXE90
     HISTORY
       DATASET-OWNER-----(NULL)   CREATION--------2013.311
       RELEASE----------------2   EXPIRATION------0000.000
     VOLUMES
       VOLSER------------MVXE90   DEVTYPE------X'3010200F'   FSEQN-------------0
     ASSOCIATIONS--------(NULL)
     ATTRIBUTES
         THE NUMBER OF ENTRIES PROCESSED WAS:
         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS
```

CAIPARM Control Statement

When performing an EXTEND or TEST, the data set that is specified by the CAIPARM DD statement must contain proper values for the following control parameters:

ALIAS=

Specify the name of the VMF Alias you want to use. You can use the same name as your primary VMF or a different name. The CATVMFX process creates the catalog entry for this alias when necessary.

F1=

Specify the name of the old VMF that is the VMF copy source. At the end of the CATVMFX EXTEND process, all TLMSs switch from this VMF to a new VMF, and this VMF is poisoned and dropped. At the end of the CATVMFX TEST process, the old VMF remains unaffected.

VMF2=

Specify the name of the new VMF that is the VMF copy destination. At the end of the CATVMFX EXTEND process, all TLMSs switch to this VMF. The VMF alias points to this data set. At the end of the CATVMFX TEST process, this VMF is dropped and poisoned.

COPYWAIT=

(Optional) Specify the interval in one-hundredths of a second between each copy operation. In the copy step of the EXTEND/TEST process, a specific number of blocks are copied from the primary VMF to the secondary VMF. The default is 15.

COPYBLKS=

(Optional) Specify the number of blocks from the primary VMF to copy during each copy operation. The default is 2400.

EXCLUDE=()

(Optional) Specify a comma-separated list of TLMS CPUs to exclude from the EXTEND/TEST process. Use this parameter carefully. EXCLUDE does not prevent the specified CPUs from writing to the VMF. It only causes CATVMFX to exclude the CPUs when it issues commands and checks for responses. Use EXCLUDE for a CPU which shows as active on the verification display but is no longer connected to this VMF. If you exclude a CPU, ensure that it is excluded in the verification display.

Procedure to Follow after CATVMFX Initial Run

After you run CATVMFX, ensure that no chaining errors were created and that the new volumes are in the new VMF.

Follow these steps:

1. Execute CAI.CTAPPROC(CATVMFV) to ensure that no critical chaining errors exist in the VMF.
2. Verify that VOLSERS added in the EXTEND display through the TLMS ISPF Interface.
3. Back up the VMF using CAI.CTAPPROC(CATVMFB).

CATVMFX Recovery Procedures

Review any error messages that CATVMFX issues. Because of the nature and impact of this utility, contact CA Support for any error conditions that occur.

Retain job logs and CAIMSG data sets for CATVMFX and all CTS/TLMSs.

CATVMFXI - Initialize or Re-create the VMF Index File

The CATVMFXI procedure initializes and creates the VMF index file from the VMF. After the VMF has been created and updated with data about your tape library and contains nonscratch tape records, you can allocate and initialize the VMF index file. A onetime process, it is executed during installation, but you can use this program anytime to rebuild the VMF index file.

The VMF index file must be recreated whenever the VMF is restored. Taking care of CI and CA splits can cause you to delete and re-create the VMF index.

Process

To recreate the VMF index, follow these steps:

1. Execute the CATVMFXI procedure with the appropriate JCL overrides and any required alterations to the IDCAMS control statements.
2. Prepare the CONVERT command as input to the VMFINDEX.SYSIN DD statement. Starting in position 1, the format of the command is

   CONVERT

CATVMFXI Procedure JCL

The following procedure initializes the VMF index file:

```
//*******************************************************
//*****             PROCNAME=CATVMFXI ****              *
//*******************************************************
//**        PROCEDURE TO CREATE A VMF INDEX FILE       *
//*******************************************************
//CATVMFXI PROC A='*',
//             LOAD='CAI.CTAPLINK',
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',
//             SORTLIB='SYS1.SORTLIB',
//             VMF='CAI.TLMS.VMF',
//             VMFIDX='CAI.TLMS.VMFINDEX',
//             WORK='SYSDA'
//*
//*******************************************************
//IDCAMS     EXEC PGM=IDCAMS
//*
//SYSPRINT   DD SYSOUT=&A
//*
//SYSIN      DD DUMMY VSAM DEFINITION
//*
//*******************************************************
//**           BUILD THE INDEX RECORDS                 *
//*******************************************************
//VMFINDEX   EXEC PGM=TLMSVSUT,
//               PARM='&PRM',
//               COND=(0,NE)
//*
//STEPLIB    DD DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS   DD DSN=&OPTS,DISP=SHR
//*
//CAIVMF     DD DSN=&VMF,DISP=SHR
//*
//CAIVMFXI   DD DSN=&VMFIDX,DISP=SHR
//*
```

```
//SORTLIB     DD DSN=&SORTLIB,DISP=SHR
//*
//SYSOUT      DD SYSOUT=&A
//*
//SYSPRINT    DD SYSOUT=&A
//*
//SORTWK01    DD UNIT=&WORK,SPACE=(CYL,(10))
//*
//SYSUDUMP    DD SYSOUT=&A
//*
//SYSIN       DD DUMMY
//*
//*************************************************************
```

Starter JCL

```
//TLMSVSUT JOB
//TLMSVSUT EXEC CATVMFXI
//*
//IDCAMS.SYSIN DD *
 DEFINE CLUSTER                    -
        (NAME(CAI.TLMS.VMFINDEX)   -
         CYL(50 50)                -
         VOL(XXXXXX)               -
         FREESPACE(20 20)          -
         INDEXED                   -
         KEYS(60 0)                -
         RECORDSIZE(100 100)       -
         SHAREOPTIONS(3 3))        -
     DATA                          -
        (NAME(CAI.TLMS.VMFI.DATA)  -
        CISZ(4096))                -
     INDEX                         -
        (NAME(CAI.TLMS.VMFI.INDEX) -
        CISZ(2048)                 -
        IMBED)
/*
//VMFINDEX.SYSIN DD *
CONVERT
/*
//
```

CATVMFXI Output

```
YOUR TLMS NN.n COMPANY NAME     TLMS VSAM INDEX CONVERSION UTILITY        TLMS057
PAGE    1
CA TLMS NN.n yymmTLrrr                                                  mm/dd/yyyy
hh.mm.ss
CONVERT
INDEX RECORDS CREATED = 0000421
```

CTSDEU - Data Erase

The CTSDEU utility erases residual data remaining on a tape following one or more valid files. By using CTSDEU, you can prevent unauthorized access to potentially sensitive data left on the end of a tape from previous use. Usually this is performed before a tape leaves a secured area. When a multivolume data set is created, every volume except the last is completely full of data. As a result, only the last volume in a multivolume dataset requires security erase processing. If a program creating multivolume datasets issues a FORCE-END-OF-VOLUME command before filling each volume, then security erase processing might be required on other volumes. In this case, specify the ALL parameter. The ALL parameter directs CTSDEU to attempt a security erase on every volume in the multivolume set.

CTSDEU Job Control Statements

```
//stepname   EXEC PGM=CTSDEU,PARM='parm'
//STEPLIB    DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSUT1     DD UNIT=(TAPE,,DEFER),DISP=(OLD,KEEP),
//           LABEL=(,SL),                              see NOTE 1
//           VOL=SER=anyvol,                           see NOTE 2
//           DSN=any.tape.file
NOTE 1: LABEL=(,BLP) is needed to erase scratch tapes.
NOTE 2: The JCL parameter VOL=SER= is required if the requested
         DSN is not in the MVS catalog.
```

Note 1: LABEL=(,BLP) can be used when trying to erase a scratch. Use LABEL=(,SL) on all other processing of CTSDEU.

Note 2: If the requested DSN is not in the MVS catalog, the JCL parameter VOL=SER= is required.

Parameter Definitions

You can specify more than one parameter. The parameters specified are processed in the order shown, not in the order of their appearance in the EXEC statement.

ERASE

Reads the tape header label and verifies the following:

- The tape is in the CA TLMS Volume Master File (VMF).
- The tape is not a virtual tape.
- The dataset names match those records in the VMF.

If the volume serial number was not supplied by VOLLABI (the VOLSER from the VOL1 record on SL or AL tapes), you must supply the volume serial number.

Note: When ERASE is used on a Scratch Tape, we assume you want to erase the whole tape. Using LABEL=(1,BLP) causes the catalog record to remain in Scratch Status. The DSN field in your tape catalog will be changed to:

SYSddddd.Tnnnnnn.RA000.CTSDEU.INITDSN where ddddd is the date that it was initialized and nnnnnn is the time of day.

NOERASE

Reads the tape header label and verifies that the tape matches the VMF, but no residual data is erased. Because the erase is not performed, your intervention is suppressed.

Note: If the NOERASE parameter is used on a scratch tape, the specified scratch tape is removed from scratch status. Only the volume specified in your JCL is mounted.

Default: ERASE

ALL

Indicates that all volumes of the requested tape file should be secured. When sets of tapes are produced, they are filled completely, overlaying any old data, but if force end of volume (FEOV) macro is used to produce short tapes, data previously written may not be overlaid.

LAST

Indicates that only the last volume of the requested tape file will be secured. All volumes of the requested file are read but only the last volume is secured.

Default: ALL

LIST

Produces a detailed list of the contents of the requested volumes.

NOLIST

A detailed list is not produced.

Default: NOLIST

Return Codes

CTSDEU can issue one of the following return codes:

0

CTSDEU processed successfully

8

Errors reading the database

12

General problems with CTSDEU (for example, parameter error, print error, and so forth)

16

Problem with the tape (for example, no write ring, database discrepancy, and so forth)

CTSTAPER - Test the Tape Management System

CTSTAPER is used to test the tape management system by issuing OPEN, CLOSE, READ, WRITE and FEOV (force end-of-volume) macros to exercise the tape functions. Up to 10 tape drives may be used per step, with control information provided in the ddname and EXEC parameter.

This utility produces the CA TLMS Tape Activity Report, and messages may be sent to the system console from an optional SYSIN data set.

The parameter value is optional and may be the value JFCB or ABEND. If omitted, the report will print each OPEN and CLOSE for a DD statement. If JFCB is specified, a hexadecimal dump of the JFCB is also printed with each OPEN and CLOSE. If ABEND is specified, CTSTAPER will terminate with an S0C3 abend after tape processing instead of a normal CLOSE. This allows testing of third disposition processing.

Processing for each tape is specified by ddname in the following format:

```
ttnnxxxx
```

Where:

tt

Is one of the following:

IN

read input

OU

write output

nn

The number of volumes to process

xxxx

User name for DD

CTSTAPER writes 24 characters twice per data set per volume, and expects to read the same data. It uses FEOV to create multivolume data sets without writing more data.

CTSTAPER can display text to the system console and also allows the job to be aborted. If the SYSIN DD statement is present, the first 70 characters of each record will be issued to the console as a WTO before any tape processing is started; a WTOR is issued at EOF, which causes a pause in the job stream. This pause allows the operator to read the text before processing is started. Additional pauses may be caused by placing *PAUSE* in positions 1-7 of a record. The WTOR issued at any pause prompts the operator for a reply of U or C; C (cancel) causes an S122 abend, and any other reply allows processing to continue.

The SYSIN data set provides the means to request detail setup and allows the operator to abort, if required.

CTSTAPER Job Control Statements

```
//TAPETEST EXEC PGM=CTSTAPER,PARM=JFCB|ABEND,[JFCB] [,ABEND]
//STEPLIB DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT DD SYSOUT=*
//TTNNXXXX DD DSN=TEST.DATASET.NAME,UNIT=TAPE,...
//SYSIN DD DATA
TEXT TO BE SENT TO THE OPERATOR CONSOLE.
```

Example:

In this example, four data sets are created spanning 3 volumes. Then these four data sets are read in the second step.

OU02SL1 causes TEST.FIRST.FILE to be written to the first and second tape volume. OU01SL2 causes TEST.SECOND.FILE to be added to the second tape volume. OU02SL3 causes TEST.THIRD.FILE to be added to the second tape volume and to span to a third. OU01SL4 causes TEST.FOURTH.FILE to be added to the third tape volume. The tape reads in the second step work the same way.

```
//TAPEOUT EXEC PGM=CTSTAPER
//STEPLIB DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT DD SYSOUT=*
//OU02SL1 DD DSN=TEST.FIRST.FILE,DISP=(NEW,CATLG),UNIT=TAPE,
//           VOL=(,RETAIN),LABEL=(1,SL)
//OU01SL2 DD DSN=TEST.SECOND.FILE,DISP=(NEW,CATLG),UNIT=TAPE,
//           VOL=(,RETAIN,REF=*.OU02SL1),LABEL=(2,SL)
//OU02SL3 DD DSN=TEST.THIRD.FILE,DISP=(NEW,CATLG),UNIT=TAPE,
//           VOL=(,RETAIN,REF=*.OU01SL2),LABEL=(3,SL)
//OU01SL4 DD DSN=TEST.FOURTH.FILE,DISP=(NEW,CATLG),UNIT=TAPE,
//           VOL=REF=*.OU02SL3,LABEL=(4,SL)
//*
//TAPEIN EXEC PGM=CTSTAPER
//STEPLIB DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT DD SYSOUT=*
//IN02SL1 DD DSN=TEST.FIRST.FILE,DISP=SHR
//IN01SL2 DD DSN=TEST.SECOND.FILE,DISP=SHR,UNIT=AFF=IN02SL1
//IN02SL3 DD DSN=TEST.THIRD.FILE,DISP=SHR,UNIT=AFF=IN02SL1
//IN01SL4 DD DSN=TEST.FOURTH.FILE,DISP=SHR,UNIT=AFF=IN02SL1
//SYSIN DD DATA
/*
//
```

CTSPMTST - Validating Pattern Definitions

The CTSPMTST utility is used to test the character strings or patterns that you have created for use in the various CAI.CTAPOPTN members or utility programs that support pattern masking. A pattern consists of a combination of alphanumeric characters and special characters that perform unique masking operations during match and compare functions for the purpose of identifying and selecting data sets. The rules for defining a pattern are discussed in this guide.

Once you have created your patterns, test them by executing the CTSPMTST utility. CTSPMTST checks each pattern for conformity to the rules of structure and syntax required by the CA TLMS Pattern Masking Facility and produces printed output to aid in resolving invalid pattern errors.

Report Description

CTSPMTST produces hardcopy output which is shown on 355.

Job Control Statements

```
//CTSPMTST  EXEC PGM=CTSPMTST,PARM='verb,class'
//STEPLIB   DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSPATT   DD *
(control statements here)
/*
//SYSOBJ   DD *          *NOTE
(control statements here)
/*
```

Note: The SYSOBJ DD statement is not required if the verb specified is VALIDATE.

Parameter Definitions

verb

Possible values for verb are:

COMPARE

Compares two patterns and indicates whether the first pattern (SYSPATT DD) is the same, more specific, or less specific than the second pattern (SYSOBJ DD).

MATCH

Matches patterns from the SYSPATT DD against objects from the SYSOBJ DD and reports the results.

VALIDATE

Tests structure and syntax and reports on the validity of the patterns pointed to by the SYSPATT DD. This parameter does not use the SYSOBJ DD.

CONVERT

Performs all the functions of the VALIDATE parameter but also returns the pattern in the internal form used by CTSPM. This parameter does not use the SYSOBJ DD statement.

class

Indicates the object class of the patterns and objects to be processed. These classes define the characteristics of the objects to be matched. Possible values for class are:

MVSFILE

z/OS 44-character data set name. This class is nodal and uses the period as a node separator.

MVSJOB

Any standard 8-byte name that can be found in the JCL. This applies to step name, program name, unit name, ddname, and job name. This class is nodeless.

JCL Considerations

SYSPATT DD

Points to one or more patterns used in processing the action requested with the verb parameter.

SYSOBJ DD

When the COMPARE verb is specified, this DD defines one or more patterns which are compared to the patterns defined in SYSPATT to determine which patterns are more specific.

When the MATCH verb is specified, this DD specifies a list of objects to be matched to the patterns specified in the SYSPATT DD. Each object specified in the SYSOBJ DD is compared to each pattern specified in the SYSPATT DD.

This DD is not used when the VALIDATE verb is specified. It is ignored if provided.

Example 1

The EDM rules defined in CTOEDMxx permit the specification of job names, DD names, program names or data set names to identify which EDM is to be assigned. If you plan to use job names to assign the EDM and would like to test the match processing before implementing the rules, execute CTSPMTST with the MATCH,MVSJOB parameter as in the following example.

Note: If program name or DD statement were to be tested that the MATCH,MVSJOB parameter would also be used.

```
//STEP1 EXEC PGM=CTSPMTST,PARM='MATCH,MVSJOB'
//STEPLIB DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSPATT DD *
MFG!!###
/*
//SYSOBJ DD *
MFGAA000
MFGA000
MFGAA00
MFGZZ000
MFGZZ999
```

Each test job name specified in the SYSOBJ DD is compared to the pattern specified in the SYSPATT DD and the results of the match are reported. The results of executing this job are displayed in Sample Output - Example 1.

Sample Output - Example 1

```
PARM: MATCH,MVSFILE
 PATTERN/OBJECT     RESULT FEED BACK MESSAGE
PATTERN: MFG!!###   NORMAL PM00-00: NORMAL COMPLETION
OBJECT: MFGAA000
PATTERN: MFG!!###   NOMATCH PM04-00: PATTERN AND OBJECT DO NOT MAT
CH
OBJECT: MFGA000
PATTERN: MFG!!###   NOMATCH PM04-00: PATTERN AND OBJECT DO NOT MAT
CH
OBJECT: MFGAA00
PATTERN: MFG!!###   NORMAL PM00-00: NORMAL COMPLETION
OBJECT: MFGZZ000
PATTERN: MFG!!###   NORMAL PM00-00: NORMAL COMPLETION
OBJECT: MFGZZ999
```

Example 2

The CTSPMTST program may be used to test data set selection processing by specifying test vault patterns in the SYSPATT DD and test data set names in the SYSOBJ DD as in the following example.

```
//STEP1 EXEC PGM=CTSPMTST,PARM='MATCH,MVSFILE'
//STEPLIB DD DSN=CAI.CTAPLINK,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSPATT DD *
ABC*.LEVEL2
ABC*.LEVEL#
/*
//SYSOBJ DD *
ABCDEF.LEVEL3
ABCDEF.LEVEL2
ABCDEF.LEVEL2.ABC
ABC.NOTHING
/*
//
```

The results of executing this job are displayed in the Sample Output - Example 2.

Sample Output - Example 2

```
THURSDAY, APRIL 8, 2004.210                  Y O U R   C O M P A N Y   N A M E                    PAGE=00001

PARM: MATCH,MVSFILE

         PATTERN/OBJECT                          RESULT  FEED BACK MESSAGE

PATTERN: ABC*.LEVEL2                           NOMATCH  PM04-00: PATTERN AND OBJECT DO NOT MATCH
OBJECT: ABCDEF.LEVEL3

PATTERN: ABC*.LEVEL2                           NORMAL   PM00-00: NORMAL COMPLETION
OBJECT: ABCDEF.LEVEL2

PATTERN: ABC*.LEVEL2                           NOMATCH  PM04-00: PATTERN AND OBJECT DO NOT MATCH
OBJECT: ABCDEF.LEVEL2.ABC

PATTERN: ABC*.LEVEL2                           NOMATCH  PM04-00: PATTERN AND OBJECT DO NOT MATCH
OBJECT: ABC.NOTHING

PATTERN: ABC*.LEVEL#                           NORMAL   PM00-00: NORMAL COMPLETION
OBJECT: ABCDEF.LEVEL3

PATTERN: ABC*.LEVEL#                           NORMAL   PM00-00: NORMAL COMPLETION
OBJECT: ABCDEF.LEVEL2

PATTERN: ABC*.LEVEL#                           NOMATCH  PM04-00: PATTERN AND OBJECT DO NOT MATCH
OBJECT: ABCDEF.LEVEL2.ABC

PATTERN: ABC*.LEVEL#                           NOMATCH  PM04-00: PATTERN AND OBJECT DO NOT MATCH
OBJECT: ABC.NOTHING
```

CTSPMTST Report Field Definitions

PARM: xxxxxxxx

Parameters passed to program

PATTERN/OBJECT

Pattern or object processed

RESULT

Result of the process

FEED BACK MESSAGE

Processing message

CTSSYNC - 3495/3494 Tape Library Data Server Synchronization Utility

CTSSYNC provides a high-level interface to the IBM 3495/3494 LCS (Library Control System) and optionally lists the results of client-requested functions. This utility performs the following functions to support the 3495/3494 device:

- Synchronizes the tape management catalog system and the 3495/3494 database. The scratch status of a volume is obtained from the tape management database and is set in the 3495/3494 LCS database.
- Ejects cartridges needed for off-site storage from the 3495/3494. Tapes are ejected to the BULK or Convenience locations.
- Changes the status of cartridges as they are scratched. The status is set in the 3495/3494 LCS database according to the specified command without reference to the tape management database.
- IMPORTs and EXPORTs virtual tapes from the 3495/3494. Virtual tapes are extracted from or added to physical container tapes that can be moved into or out of the 3495/3494 robot.

This utility accepts input from either a PARM value or SYSIN control statement. The PARM value can be either from a calling program or the EXEC JCL statement. Commands specified without VOLSERs are global commands that are used for subsequent VOLSERs specified without commands. Global commands are replaced by subsequent global commands, but not specific commands which include a VOLSER. The default global command is SYNC. Each command is processed when the VOLSER or VOLSER range is specified regardless of any previous or subsequent commands.

Job Control Statements

```
//stepname EXEC PGM=CTSSYNC,PARM='parm'
//STEPLIB DD DISP=SHR,DSN=CAI.CTAPLINK
//SYSPRINT DD SYSOUT=*     *NOTE 1
//ERREPORT DD SYSOUT=*     *NOTE 2
//EXPORT DD DSN=EXPORT.TAPE.LIST,   *NOTE 3 *NOTE 6
  UNIT=VTAPE,TRTCH=NOCOMP,DISP=(,CATLG)
//IMPORT  DD DSN=IMPORT.TAPE.LIST,   *NOTE 4 *NOTE 6
  UNIT=VTAPE,TRTCH=NOCOMP,DISP=(,CATLG)
//SYSIN DD *      *NOTE 5
(control statements here)
/*
```

Note 1: The SYSPRINT DD is optional. If coded, CTSSYNC writes the standard (or full) report to this file. The standard report produces a detail list of the requested commands for each volume and the results of those requests. When the command is for a range of volumes a line is listed for each volume. For SYNC commands the results of the tape management query is also shown.

Note 2: The ERREPORT DD is optional. If coded, an errors-only report is produced. This report is in same format as the standard report.

Note 3: The EXPORT DD is required for use by the EXPORT command. CTSSYNC writes all the files required by the 3494/VTS for an EXPORT operation to the virtual tape defined by this DD statement. If no export commands are specified, then this DD is NOT used. Do not specify 'VOL=(,RETAIN)' on this DD statement. See Note 6.

Note 4: The IMPORT DD is required for use by the IMPORT command. CTSSYNC writes all the files required by the 3494/VTS for an IMPORT operation to the virtual tape defined by this DD statement. If no import commands are specified, then this DD is NOT used. Do not specify 'VOL=(,RETAIN)' on this DD statement. See Note 6.

Note 5: The SYSIN DD is optional. The commands can be entered by using the SYSIN control statements, or by coding them as parameters after the PARM= statement, or by passing a parameter list with the use of register 1. Commands are free-form, but the entire command line must be on a single line between columns 1 and 71. Spaces and commas are delimiters. Comments must have the form /* comment text */. They can appear anywhere within a command or on a line by themselves.

Note 6: Do not specify 'VOL=(,RETAIN)' on this DD statement. Specifying VOL=(,RETAIN) will cause the IMPORT/EXPORT list volume to remain on the tape drive. The robot does not consider the list volume available if it is mounted, so the IMPORT/EXPORT invoked by CTSSYNC will fail.

Note 7: The CTSSYNC utility cannot be used with 3495/3494 BTLS devices.

Parameter Definitions

The CTSSYNC utility processes the following commands:

```
VERIFY
NOVERIFY
BULK
NOBULK
FORCE
NOFORCE
EJECT{,vvvvvv{-vvvvvv}}
PROTECT{,vvvvvv{-vvvvvv}}
SCRATCH{,vvvvvv{-vvvvvv}}
PRIVATE{,vvvvvv{-vvvvvv}}
SYNC{,vvvvvv{-vvvvvv}}
EXPORT{,llllll{-llllll}} {,destination}
IMPORT{,pppppp} {,llllll{-llllll}} {,ooooooooo}
```

Where:

vvvvvv

volume serial number (real or virtual)

llllll

volume serial number for a logical (virtual) tape

pppppp

volume serial number for a physical (container) tape

ooooooo

IMPORT option either SCRATCH or PRIVATE

VERIFY/NOVERIFY

The VERIFY command instructs the system to verify that the volume serial number is defined in the 3495/3494 LCS database prior to executing any further commands. If NOVERIFY is specified, the LCS commands are issued even if the cartridge does not reside in the 3495/3494 LCS database. VERIFY is the default.

SYNCTEST[,vvvvvv[-vvvvvv]]

COPY,POOL[,pp] where pp is the physical volume pool that contains the logical volumes to export.

BULK/NOBULK

The BULK command instructs any following EJECT commands to be sent to the bulk location. The NOBULK command instructs all following EJECT commands to be sent to the convenience location. If neither BULK nor NOBULK was specified prior to an EJECT command, the utility defaults to the bulk location.

FORCE|NOFORCE

The FORCE command instructs the program to execute any following SYNC commands unconditionally, regardless of the current use attribute in the TCDB. The NOFORCE command causes SYNC commands to be processed only if the use attribute in the TCDB does not match the volume status in the tape database, or if TCDB or Library Manager show an error indicator. NOFORCE is the default.

EJECT

The EJECT command instructs the 3495/3494 robotic device to eject the specified VOLSERs. Depending upon whether a BULK or NOBULK command was issued prior to the EJECT command, the cartridge is ejected to either the bulk or the convenience location.

PROTECT/SCRATCH/PRIVATE

The PROTECT, SCRATCH, and PRIVATE commands all change the status of the specified VOLSERs. The SCRATCH command changes the status to scratch, while the PROTECT and PRIVATE commands change the status to private within the 3495/3494 LCS database. This affects which cartridges are selected to satisfy SCRATCH requests and which cartridges are added to any auto cartridge loaders (ACLs) located inside the 3495/3494 robotic device. There is no difference between PROTECT and PRIVATE.

SYNC

The SYNC command causes the specified VOLSERs to be synchronized between the tape management database and the 3495/3494 LCS database. If the status is different, the tape management database is assumed to be correct. This is the default if no specific command is specified.

EXPORT

The EXPORT command causes the specified LOGICAL VOLSERs to be written to the export list volume (a logical volume residing in the same library as the volumes to be exported). After all volumes have been written to the export list volume, the OAM LCS schedules the list to be exported.

DESTINATION

The DESTINATION sub-parameter can only be included on the EXPORT control statement and is optional. If included, all virtual volumes with the same destination will be stacked together on a single physical volume. If a list of 50 virtual volumes to be exported had been supplied; and 25 had one destination and the other 25 had a different destination, then the first 25 would be stacked together on one set of physical volumes and the other 25 would be stacked on a different set of physical volumes. However, this value is not sent to the CBRUXEJC exit during eject processing nor is it tracked in the TCDB or the tape management catalog.

IMPORT

The IMPORT command causes the specified LOGICAL VOLSERs to be written to the import list volume; that is, a logical volume residing in the same library as the volumes to be imported. After all volumes have been written to the import list volume, the OAM LCS schedules the list to be imported. Only 1 EXPORT or IMPORT can be scheduled at a time. Subsequent EXPORTs or IMPORTs will be shown as errors by CTSSYNC.

SCRATCH/PRIVATE

SCRATCH and PRIVATE are optional sub-parameters that can be included on the IMPORT control statement if the specific virtual volumes are specified. PRIVATE, the default, indicates that the specified virtual volume should be copied back into cache inside the 3494/VTS completely; that is, the virtual volume is copied. If SCRATCH is specified, then the specified virtual volume is assumed to be in scratch status and only the VOL1/HDR1/HDR2 records are copied back into cache. While this reduces the time to complete the IMPORT process, those virtual volumes cannot be made un-scratched because the data on them no longer resides inside the 3494/VTS.

ABDLMT

The ABDLMT keyword parameter is used to set a return code limit at which CTSSYNC will ABEND. After CTSSYNC has completed all other processing and is about to terminate, the numeric value of ABDLMT is compared with the highest return code from any command. If that return code is greater than or equal to the ABDLMT, CTSSYNC abends with a U0006 and a reason code equal to the return code. The default is ABDLMT=99. ABDLMT can be specified:

- In either the ABDLMT=nn or the ABDLMT(nn) form
- In the EXEC parm field or in a SYSIN statement
- On a statement alone or with any command
- Multiple times, but only the last value is used

SYNCTEST

Performs like the SYNC command but does not call to update the OAM. Also, both the TCDB (VOLCAT) and the Library Manager (LM) database are compared with the TMC. If the status of the volume is not synchronized in all three data bases or catalogs, an error is reported to both the SYSPRINT and ERREPORT DD statements.

COPY

Performs like the EXPORT command except that the exported volumes are copied, not moved. List the VOLSERs to be exported below the COPY command in the SYSIN DD.

Note: Specify the POOL parameter before you export the volumes.

POOL

Specifies the physical volume pool that contains the logical volumes to export. This command is only used with the COPY command.

Examples of Specifying Commands to CTSSYNC

There are three methods of getting the commands or VOLSERs to the CTSSYNC utility: PARM= on the EXEC statement, through SYSIN control statements, or passed in a parameter list through register 1. The easiest way to perform a single command is through use of the PARM statement on the EXEC command.

```
//STEP1 EXEC PGM=CTSSYNC,PARM='EJECT,123456'
```

This causes cartridge 123456 to be ejected from the 3495/3494 robotics device. Likewise, the same value could have been used as input from the SYSIN control statements.

```
//SYSIN  DD *

EJECT,123456
```

The command can also be passed in a parameter list pointed to by register 1. This would be used if another program; for example, the security exit, wanted to call the synchronization program passing it the command. The format of the parameter list pointed to by register 1 is as follows:

```
R1 ==> A(addr1) ==> CL8'EJECT'
       A(addr2) ==> CL8'123456'
```

It is possible to use the SYSIN control statement in addition to either of the two parameter methods. For example;

```
//STEP1  EXEC PGM=CTSSYNC,PARM='BULK'
//SYSIN  DD *
EJECT,123456
EJECT,100001
EJECT,001234
```

In this example, the BULK command applies to all subsequent commands. Therefore, the EJECT of tapes 123456, 100001, and 001234 would be directed to the bulk location. If a volume command is issued without a VOLSER, all subsequent VOLSERs would then be processed.

```
//STEP1 EXEC PGM=CTSSYNC
//SYSIN DD *
SYNC
A00001-A00100
100002
102030
100202
```

In this example, this range and the three listed volumes (100002, 102030, and 100202) would have their statuses synchronized between the tape management system and the 3495/3494 LCS database. This could just as easily be a list of tapes to be ejected to the BULK location (as a post process to the vaulting system for example).

```
//STEP1 EXEC PGM=CTSSYNC
//EXPORT DD DSN=EXPORT.TAPE.LIST,
// UNIT=VTAPE,TRTCH=NOCOMP,DISP=(,CATLG)
//SYSIN DD *
EXPORT,LV1234,OFFSITE1 /* comments */
EXPORT,LV2345,OFFSITE1
```

In this example, the logical volumes LV1234 and LV2345 are to be removed. An export list volume is created and the 3495/3494 LCS schedules the list to be exported. When the export function is completed, logical volumes LV1234 and LV2345 are stacked on to a physical volume and the physical volume is put in export hold status, waiting to be removed from the 3495/3494.

```
//STEP1 EXEC PGM=CTSSYNC
//IMPORT DD DSN=IMPORT.TAPE.LIST,
// UNIT=VTAPE,TRTCH=NOCOMP,DISP=(,CATLG)
//SYSIN DD *
IMPORT,PV0001,LV1234
IMPORT,PV0001,LV2345-LV2348 /* range of logical volumes */
```

In this example, logical volumes LV1234 and LV2345 thru LV2348 are imported from physical volume PV0001. An import list volume is created and the 3495/3494 LCS schedules the list to be imported.

```
//STEP1 EXEC PGM=CTSSYNC
//IMPORT DD DSN=IMPORT.TAPE.LIST,
// UNIT=VTAPE,TRTCH=NOCOMP,DISP=(,CATLG)
//SYSIN DD *
IMPORT,PV0001
```

In this example, all logical volumes on physical volume PV0001 will be imported. An import list volume is created and the 3495/3494 LCS schedules the list to be imported.

The CTE3495 member in CAI.CTAPEARL can be used to help generate a list of VOLSERs as input to CTSSYNC.

```
//STEP1 EXEC PGM=CTSSYNC,PARM='SCRATCH,ABDLMT=12'
//IMPORT DD DSN=IMPORT.TAPE.LIST,
// UNIT=VTAPE,TRTCH=NOCOMP,DISP=(,CATLG)
//SYSIN DD *
012345 /* SINGLE TAPE WITH COMMENT */
ABC017-ABC052 /* SCRATCH ALL TAPES IN THIS RANGE */
SYNC XYZ100-XYZ299
```

In this example, the EXEC PARM is used to set 'SCRATCH' as a global command and to ABEND CTSSYNC if the highest return code is greater than or equal to 12. The VOLSER 012345 and the range of VOLSERs ABC017 through ABC052 will be scratched through the global command. The range of VOLSERs XYZ100 thru XYZ299 will be synchronized with the tape management system.

Completion Codes

Possible completion codes and their meaning are listed below.

Codes	Explanation
0	Successful completion. The requested commands were performed successfully. This code can also indicate that requested commands were not processed because no VOLSER was provided.
4	Successful completion with redundant commands. In VERIFY mode, one or more ATL commands failed or could not be performed because the VOLSER was not defined in the 3495/3494 LCS database. For the SYNC command, this code can also indicate synchronization of one or more volumes was bypassed, due to their status in the tape management database. For the PRIVATE, PROTECT or SCRATCH command, this code can also indicate that the command was issued for a volume which already was in the requested status in the LCS database.
8	Processing errors. One or more ATL commands failed or could not be performed because the VOLSER was not defined in the 3495/3494 LCS database, and NOVERIFY was specified. This code can also indicate the following conditions: ■ One or more ATL commands failed for other reasons ■ No valid input command was found ■ The tape management system was not active
12	Incorrect job setup. A required DD statement was missing.
16	Invalid request. A schedule of both IMPORT and EXPORT operations was attempted.

CTSSYNC ATL Response Log Report

```
OCT10,2004 9:54:12                                                    CTSSYNC ATL RESPONSE LOG                  PAGE
        1
                  FUNCTION ---------------------- ATL ACTION ---------------------------- STATUS IN TAPE DATABASE -----------
                  PROTECT TDI001                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  EJECT TDI002                            EJECT COMPLETE
                  SCRATCH TDI003                          SYNC PROCESS COMPLETE
                  PROTECT TDI004                          SYNC PROCESS COMPLETE
                  PROTECT TDI005                          SYNC PROCESS COMPLETE
                  PROTECT TDI006                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  VERIFY                                  VERIFY MODE INITIATED
                  NOVERIFY                                VERIFY MODE TERMINATED
                  BULK                                    BULK-EJECT DOOR ACTIVE
                  NOBULK                                  CONVENTIENCE DOOR ACTIVE
                  PROTECT TDI016                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  PROTECT TDI017                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  PROTECT TDI018                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  PROTECT TDI019                          SYNC PROCESS COMPLETE                    VOLUME NOT IN SCRATCH STATUS
                  SCRATCH TDI020                          SYNC PROCESS COMPLETE                    VOLUME IN LIBRARY AND SCRATCH
                  PROTECT TDI021                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  PROTECT TDI022                          SYNC PROCESS COMPLETE                    EDM/DYNAM CONTROLLED
                  PROTECT TDI028                          SYNC PROCESS COMPLETE                    VOLUME NOT IN SCRATCH STATUS
                  PROTECT TDI029                          SYNC PROCESS COMPLETE                    VOLUME NOT IN SCRATCH STATUS
                  SCRATCH TDI030                          SYNC PROCESS COMPLETE                    VOLUME IN LIBRARY AND SCRATCH
                  SCRATCH TDI031                          SYNC PROCESS COMPLETE                    VOLUME IN LIBRARY AND SCRATCH
                  SCRATCH TDI032                          SYNC PROCESS COMPLETE                    VOLUME IN LIBRARY AND SCRATCH
ERROR(S)   SYNC ABC001                                    ATL SYNCRONIZE BYPASSED
        VOLUME NOT DEFINED TO THE DATABASE
ROR(S)     SYNC ABC002                                    ATL SYNCRONIZE BYPASSED
        VOLUME NOT DEFINED TO THE DATABASE
ERROR(S)   SYNC ABC003                                    ATL SYNCRONIZE BYPASSED
        VOLUME NOT DEFINED TO THE DATABASE
ERROR(S)   SYNC ABC004                                    ATL SYNCRONIZE BYPASSED
        VOLUME NOT DEFINED TO THE DATABASE
ERROR(S)   SYNC ABC005                                    ATL SYNCRONIZE BYPASSED
        VOLUME NOT DEFINED TO THE DATABASE
                  SCRATCH TDI001                          SYNC PROCESS COMPLETE
                  SCRATCH TDI001                          VOLUME IS ALREADY IN THAT STATUS
                  SCRATCH TDI001                          VOLUME NOT FOUND BY ATL
ERROR(S)   SCRATCH TDI001                          REQUEST FAILED FROM LCS
                  IMPORT TDI003      FROM XMIT10          WRITTEN TO IMPORT LIST VSN MVXE9                    0
                  IMPORT TDI004      FROM XMIT10          WRITTEN TO IMPORT LIST VSN MVXE9                    0
                  IMPORT TDI005      FROM XMIT10          WRITTEN TO IMPORT LIST VSN MVXE90
ERROR(S)   EXPORT TDI006                           BYPASSED MIXING IMPORT AND EXPORT
ERROR(S)   EXPORT TDI007                           BYPASSED MIXING IMPORT AND EXPORT

               QUERY   DC0011                QUERY COMPLETE, VOLUME IN ROBOT

               SYNC    DC0011                VOLUME WAS ALREADY IN THAT STATUS        VOLUME NOT IN SCRATCH STATUS

               QUERY   DC0012                QUERY COMPLETE, VOLUME IN ROBOT

               SYNC    DC0012                VOLUME WAS ALREADY IN THAT STATUS        VOLUME NOT IN SCRATCH STATUS

               QUERY   DC0013                QUERY COMPLETE, VOLUME IN ROBOT

ERROR(S)       QUERY   DC0013                OUT OF SYNC: TVIUSEA = S & TVILCAT = UN


                                                                  ---- END OF REPORT ----
```

CTSSYNC Report Field Definitions

FUNCTION

The function, command, or parameter that was passed to CTSSYNC. A SYNC command is broken down first into a QUERY type transaction (to find out if the cartridge is in the robot or not and what its status is if it is inside the robot) and then either a PROTECT command (to change its status from scratch to private), a SCRATCH command (to change its status from private to scratch) or a SYNC command (meaning that it was already in the correct status).

ATL ACTION

A description of what the ATL has been asked to do and what its response was.

STATUS IN TAPE DATABASE

A description of the status of the cartridge in the tape management database. This is used only for a SYNC command.

ATL ACTION Messages

The following ATL ACTION messages are issued.

VERIFY MODE INITIATED

The VERIFY command had been issued and processed.

VERIFY MODE TERMINATED

The NOVERIFY command had been issued and processed.

CONVENIENCE DOOR ACTIVE

The NOBULK command was issued and the convenience door will be used for any further EJECT commands.

BULK-EJECT DOOR ACTIVE

The BULK command was issued and the bulk door will be used for any further EJECT commands.

NO DD STATEMENT FOR IMPORT

An IMPORT command was issued, but the JCL did not contain an IMPORT DD statement.

NO DD STATEMENT FOR EXPORT

An EXPORT command was issued, but the JCL did not contain an EXPORT DD statement.

WRITTEN TO EXPORT LIST VSN vvvvvv

An EXPORT command was issued and the volume to be exported was written to the export list created on virtual-volume vvvvvv.

WRITTEN TO IMPORT LIST VSN vvvvvv

An IMPORT command was issued and the volume to be imported was written to the import list created on virtual-volume vvvvvv.

QUERY COMPLETE, VOLUME IN ROBOT

A SYNC command was issued and the volume to be synchronized did reside inside the robot.

VOLUME WAS ALREADY IN THAT STATUS

A SYNC command has been issued, and the volume was already in the correct status.

REQUESTED PROCESS COMPLETE

Either a PROTECT, SCRATCH, PRIVATE command had been issued and the volume had its status successfully changed. The PROTECT or SCRATCH command may also have been issued because of a SYNC command when the initial QUERY found the tape was not in the correct status.

EXPORT FAILED, REASON CODE=nnn

The EXPORT command failed. The reason codes can be found in the LCS External Services Parameter List (LCSPL), mapped by IBM macro CBRLCSPL. This macro can be found in SYS1.MACLIB.

IMPORT FAILED, REASON CODE=nnn

The IMPORT command failed. The reason codes can be found in the LCS External Services Parameter List (LCSPL), mapped by IBM macro CBRLCSPL. This macro can be found in SYS1.MACLIB.

REQUEST FAILED RC=rc,RS=rsn,FB=nnnnnn

The request (SCRATCH, PRIVATE, PROTECT, SYNC) failed with a return code of rc; and reason code of rs. The return code and reason code combinations can be found in the LCS External Services Parameter List (LCSPL), mapped by IBM macro CBRLCSPL. This macro can be found in SYS1.MACLIB.

SYNTAX ERROR DURING PARSE

There was a syntax error on the specified control statement.

PARSING ERROR IN THE VSN RANGE

A range of volumes had been specified, but there was a syntax error in the format.

BYPASSED MIXING IMPORT AND EXPORT

It is not allowed to perform both an IMPORT and an EXPORT on the same execution.

NO COMMAND(S) FOUND

No command was found on the PARM or in the SYSIN.

INVALID REQUEST FOR THIS ATL

A command was rejected as un-supported by the ATL software.

ATL INACTIVE OR NOT AVAILABLE

The 3494/ATL was either not active or not available to communicate with CTSSYNC.

ATL SYNCRONIZE BYPASSED

The SYNC command was bypassed for the specified reason.

VOLUME NOT FOUND BY ATL

The volume specified on the command is not defined in the ATL TCDB.

OUT OF SYNC: TVIUSEA = x & TVILCAT = yy

SYNC operation failed. The ATL TCDB status x and LM status yy are different. Use IBM ISPF panels to resolve the discrepancy.

For more information about status settings, see the IBM document DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Tape Libraries.

OUT OF SYNC: ROBOT= xxx & TAPEMGMT= yyy

The tape robot and tape management system both have different settings. Running CTSSYNC with a PARM='SYNC' fixes this.

Status in TAPE DATABASE Messages

The following STATUS IN TAPEDATABASE messages are issued.

TAPE FULL OR EDM CONTROLLED

The requested volume was either full (it is part of a multi-volume chain) or is controlled by an external data manager. In either case, it cannot be MOD'ed onto.

VOLUME NOT IN SCRATCH STATUS

The requested volume is active (not in scratch status), but could have additional data MOD'ed onto it or have additional files stacked upon it.

VOLUME IN LIBRARY AND SCRATCH

The requested volume is in scratch status

VOLUME NOT DEFINED TO THE DATABASE

The volume is not defined to the tape management system.

TAPE MANAGEMENT IS NOT ACTIVE

The tape management system is not currently active.

TAPE CREATED WITHIN THE LAST 5 MINUTES

A SYNC command was issued for a volume that was created within the last 5-minutes, this volume was bypassed.

BAD TAPE OR OUT-OF-SERVICE

A SYNC command was issued for a tape that is either marked as BAD or as OUT-OF-SERVICE in the tape management database.

TLMSNITT - Initialize Tape Volumes

TLMSNITT initializes tape volumes as SL, NL, or AL. TLMSNITT checks the VMF and the tape label to verify your request to initialize a tape volume.

Important! Nonlabeled tapes and tapes not written to can still be initialized.

Standard OPEN and CLOSE are issued to process the tapes. SMS, CA Top Secret Security, RACF, CA TLMS, and any other components or products receiving control during OPEN and CLOSE processing may be invoked.

For any tape cartridges where a VOL1 record is present (this includes AL and SL labeled tapes), the volser is obtained from the tape. For round tapes, the VOLSER can be obtained only when the tape is known to contain data (OLDTAPE), since any attempt to read a blank tape causes the drive to search the entire reel for data. In all cases (except where the hardware indicates the tape contains no data), when a VOLSER cannot be obtained from the tape, you are prompted to provide one. To determine if the mounted tape is valid for initialization, the status of the obtained VOLSER in the VMF, the input command (INTAPE, NEWTAPE, OLDTAPE or OUTTAPE), and user exit (if any) will be checked.

TLMSNITT opens tapes with BLP input. Some security packages software protect the device if the user does not also have BLP write authority to issue a CCW = x'7782??80'. This CCW causes the device to treat the tape as write protected resulting in a write protect error message from TLMSNITT. In the case of when the user does not have BLP read authority, the system (without any indication or warning) treats BLP as NL. This can manifest itself either by successive mount/dismounts when the tape is blank or unreadable or in CAT9702P messages. TLMSNITT must be executed under a USERID which has read/write BLP authority.

The method of volume verification for TLMSNITT differs depending on the request and type of device. The overriding rule is that the operator is prompted any time the TLMSNITT cannot obtain sufficient information from the device to ensure that a 'live' tape is not being over written. The procedure differs depending upon whether the tape is being opened for input or output. Cartridges are always opened for input; reels are opened for output unless OLDTAPE is specified (opening up a blank reel for input will scan off the end of tape for data).

When a tape is opened for output with BLP, TLMS intercepts the open and issues a CAT9702P WTOR. It prompts you for the volser of the mounted tape to ensure that a *live* tape is not over written. Once a tape is opened for output, closing it results in an end-of-file mark which destroys data. When OLDTAPE is specified for a reel, you indicate that the tape contains data; thus, the tape can be opened for input and the software can attempt to read an 80 character VOL1 record from the tape. If this read is successful, then the software determines the true volser of the tape and the VMF is checked to ensure that this is not a live tape. If there is no VOL1 record (NL) on the tape, TLMS issues a CATLTN01 WTOR. When a cartridge is processed, the device recovers from attempting to read a blank tape (new, not erased) and provides sense data from the device to indicate that this is a blank tape not a live tape and issues no verification message. In most cases, when attempting to read a valid tape which cannot be processed by this device (such as a 3480 with a 36 track tape), the device returns the volser in the format 24 sense data and the VMF can be checked using this volser without your intervention. In all other cases when an error occurs attempting to read the VOL1 record (such as may be the case with a bulk erased tape) or no VOL1 record is present, TLMS issues a CATLTN01 WTOR.

TLMSNITT uses both BLP and 98000 processing. Therefore, the proper security access must be set up to run TLMSNITT. When the process updates a VMF record, the data set name and other information appears as if a temporary data set was written by program TLMSNITT. This always leaves the volume in scratch status and available for reuse.

CA TLMS must be active to run TLMSNITT.

User Exit

An optional user exit TLMSXNIT is provided for tailoring TLMSNITT processing. TLMSXNIT can continue processing, reject the tape, or modify the VMF record or tape labels.

Report Description

TLMSNITT generates a Tape Initialization. For a sample this report, see TLMS - Tape Initialization (see page 291).

Job Control Statements

```
//stepname EXEC PGM=TLMSNITT
// [,PARM='[DATEFMT=(fmt)][,TEST][,VERIFY=(opt)]']
//STEPLIB DD DSN=CAI.CTAPLINK,DISP=SHR
//TAPEDB DD DSN=VMF NAME
//SYSPRINT DD SYSOUT=A
//LABELDD DD UNIT=(tape,,DEFER)
//SYSIN DD *
(control statements here)
/*
```

Parameter Definitions

DATEFMT=(fmt)

(Optional) Specifies the date format to be printed on TLMS Tape Initialization Report. The date pattern indicated by (fmt) is limited to 10 bytes and must be enclosed in parentheses.

TEST

The TEST parameter lets you simulate the execution of TLMSNITT without modifying the contents of the tapes or the VMF. This can be done with or without operator intervention. Operator intervention is determined by the presence of a DD card for LABELDD. (In round tapes, the operator intervention can occur only when the OLDTAPE command is requested. This is because of problems opening a blank round tape for input.) When there is no LABELDD DD statement present, no operator intervention is required and console messages are kept to a minimum. Control statements are read, checked for syntax and valid operands. The SER operand is checked against the contents of the VMF (except when OUTTAPE is specified), and the user exit is called. When the LABELDD DD statement is present in the JCL, the requested tapes are actually mounted (except for round tapes which are not mounted unless OLDTAPE is specified) and additional edits are performed using the label information contained on the tape. All console messages including those requiring responses are issued.

VERIFY=(opt)

The VERIFY parameter determines when, and if, label verification messages are issued for cartridges. It is valid for all cartridges, ignored for reels unless the OLDTAPE command is used, and not applicable when the test option is specified without a DD card for LABELDD. The verify command can have one of the following values:

ABEND

Task will abend after issuing a CATLTN02 or CATLTN03 message without waiting for an operator response.

ALWAYS

A CATLTN01 message is always issued. Reels will always display '******' for the VOL1 record's volser if OLDTAPE is not specified. For example, no attempt is made to read the VOL1 record. New tapes will display '*NEW*' for the VOL1 record's volser (cartridges only, OLDTAPE is not valid for new reels and will run the tape off the end of the reel if specified for a never used tape).

CHANGE

The CATLTN01 message is issued whenever a tape containing data is having its volser changed or for tapes not containing a VOL1 record (NL), *NL*.

ERROR

Current methodology - the CATLTN01 message is issued only for tapes not containing a VOL1 record (NL).

ROBOT

No operator verify is asked for or required. Prompting is suppressed because ATLS can read the external volser.

JCL Considerations

LABELDD

Defines the tape drive that will be used to initialize volumes.

tape

Specifies a generic tape device used to initialize tape volumes. The DEFER keyword is required.

Completion Codes

Possible completion codes and their meanings are listed below.

Code	Explanation
0	Successful completion.

Code	Explanation
4	Successful completion with operator intervention. For example, a 'live' tape was mounted for initialization, it was rejected and a valid tape was mounted and successfully initialized. Another example would be a write-protected tape was mounted and rejected, or a non-write-protected tape was mounted and successfully initialized. You should check the console log for specifics.
8	All tapes were not successfully initialized. For example, a specific SER was not in the VMF or had the wrong status. You should check the TLMS Tape Initialization report (and possibly the console log) for specifics.
12	Invalid input, such as a syntax error or invalid JCL parameter. You should check the TLMS Tape Initialization report.
16	Fatal error, program cannot continue.

Note: The return code will be the highest code encountered, and does not preclude the existence of errors having lower return codes, (RC=12 does not mean that there were no errors that would not have produced an RC=8). For example, if you receive an RC=8 you should also check the console for RC=4.

Control Statement Specification

The control statement provides the following information:

- Indicates if the volume is under CA TLMS control
- Specifies the volume serial numbers for initialization
- Describes the status of the volume in the VMF

The control statements are free-form and can span to as many statements as required but cannot exceed position 71. Comments in the form /*xxxxxxx*/ can appear in the control statements.

```
verb SER=vvvvvv [,OWNER='ccccccccc|ccccccccccccccc'
    [,NUMBTAPE=n]
    [,LABTYPE=cc]
    [,DISP=REWIND|UNLOAD]
    [,ACCESS=x]
```

Control Statement Definition

verb

Specifies if the volume serial number is under CA TLMS control.

The values for verb are:

INTAPE

Specifies that the volume serial number to be initialized is under CA TLMS control and the VMF record for the volume is in scratch status. This function will ask for verification when VOLSERs differ.

NEWTAPE

Specifies that the volume serial number is under CA TLMS control and the physical tape volume is to be replaced. The VMF record for the volume must be in scratch status.

OLDTAPE

Specifies that the volume serial number is under CA TLMS control and the tape volume needs to be reinitialized. The VMF record for the volume must be in scratch status.

NUMBTAPE

If specified, must be one. OLDTAPE is the only verb which causes TLMSNITT to read the label of a reel type tape.

OUTTAPE

Specifies that the volume serial number to be initialized is not a CA TLMS controlled tape volume. When the OUTTAPE verb is used to initialize a tape volume, a log record is written to reflect the volume initialized.

Note: Only one tape is initialized for this verb regardless of NUMBTAPE (see description of NUMBTAPE).

SER=vvvvvv

Specifies the volume serial number of the first or only tape to be labeled. The value of vvvvvv can be incremented by one for each additional tape to be initialized. When processing multiple tapes, TLMSNITT starts with the specified SER= keyword and continues with the next volume that appears in the VMF for the NUMBTAPE volume. Volumes which are not in the correct status are reported and skipped. If the volume serial number is less than six characters and numeric, it must be enclosed in single quotes.

OWNER='ccccccccc[cccc]'

Specifies the owner's name or similar identification. The information is specified in character format. It can be up to 10 characters in length for EBCDIC and BCD volume labels, or up to 14 characters in length for volume labels written in ASCII. If this name contains blanks or other special characters (except periods or hyphens), it must be enclosed in single quotes. Imbedded quotes may be indicated by specifying two single quotes. If not specified, the VENDOR field from the VMF will be used. If specified, the first eight characters will be saved in the VMF VENDOR field.

NUMBTAPE=n

The value of n may be from 1 to 255. If omitted, one tape will be initialized. Otherwise initialization will be attempted of each tape in the VMF, starting with the one specified by SER= and continuing for n tapes. n will include tapes that are invalid or fail.

Note: If OUTTAPE is specified, only one tape is initialized regardless of NUMBTAPE.

LABTYPE=xx

Specifies the type of LABEL= to be written to the tape. SL is IBM standard label, AL is ANSI standard label, and NL is no label (tapemark written).

DISP=

Specifies whether to UNLOAD or REWIND the tape after initialization. UNLOAD is the default.

ACCESS=x

Single character for ANSI security access. Valid values are blank or A thru Z. This is valid only if LABTYPE= has been specified as AL.

Control Statement Examples

The following control statement will initialize a non-VMF tape with volume serial number 012345:

```
OUTTAPE SER=012345
```

To initialize five new tapes with the first five available (deleted) VMF volume serial numbers, use the following:

```
INTAPE NUMBTAPE=5,SER=012345
```

Note: In a z/OS or MSP environment, APF authorization is required for TLMSNITT and the modules it calls: CTSPARSE, CTSSIO, and CTSTXIO.

TLMS - Tape Initialization

```
TLMS                                          TLMSNITT INITIALIZATION REPORT                                        yyyy/ddd
NEWTAPE SER=TST001,DISP=REWIND,OWNER='TAPE TESTERS'
VOLUME=TST001   LABEL-TYPE=SL   ACCESS=           OWNER=TAPE TESTERS            TAPE INITIALIZED

OLDTAPE SER=TST002,DISP=REWIND,OWNER='TAPE TESTERS',NUMBTAPE=4
        WAS 980046 +
VOLUME=TST002   LABEL-TYPE=SL   ACCESS=           OWNER=TAPE TESTERS            TAPE INITIALIZED
VOLUME=TST003   LABEL-TYPE=SL   ACCESS=           OWNER=TAPE TESTERS            TAPE INITIALIZED
VOLUME=TST004   LABEL-TYPE=SL   ACCESS=           OWNER=TAPE TESTERS            TAPE INITIALIZED
VOLUME=TST005   LABEL-TYPE=SL   ACCESS=           OWNER=TAPE TESTERS            TAPE INITIALIZED

OUTTAPE SER=X00001,DISP=REWIND
VOLUME=X00001   LABEL-TYPE=SL   ACCESS=                                                 TAPE INITIALIZED
        WAS 980046

INTAPE SER=TST006,DISP=REWIND,OWNER='TAPE TESTERS'
*** TST006 ---- INITIALIZATION BYPASSED ***                                            TAPE MUST BE IN SCRATCH STATUS
```

TLMSNITT Report Field Definitions

Control statement image

Input control statement

VOLUME=xxxxxx

Volume serial number of tape to be reinitialized

LABEL-TYPE=xx

Type of label to be written on tape

ACCESS=x

ANSI security access code to be written on tape

OWNER=xxxxxxxx

Owner information to be written on tape

Message

Processing message for initialization requests

CTSTMAP – Tape Map Utility

The Tape Map (TMAP) task provides a convenient way to list the contents of an ANSI Label (AL) or Standard Label (SL) tape. It dynamically allocates one or more tapes drives, and mounts the requested volumes to produce a report of the files on each AL or SL tape. The tape drive is released after the requested tapes are processed.

CTSTMAP JCL

```
//CTSJTMAP JOB (ACCT),'CA COMMON TAPE SYSTEM'
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
//* CTSTMAP WILL DYNAMICALLY ALLOCATE A TAPE DRIVE AND SYSOUT.  *//
//*  //TAPE    DD UNIT=????,VOL=SER=??????                      *//
//*  //TAPEMAP DD SYSOUT=A                                      *//
//*                                                            *//
//* CTSTMAP WILL PROCESS FROM 1 TO 8 TAPE VSNS SEPERATED BY    *//
//*   COMMAS OR BLANKS                                         *//
//*                                                            *//
//* THE UNIT PARAMETER MAY SPECIFY A GROUP (ESOTERIC) NAME, A  *//
//* DEVICE TYPE (GENERIC) OR A SPECIFIC DEVICE ADDRESS.  TO    *//
//* CODE A 4-DIGIT SPECIFIC DEVICE ADDRESS, PRECEDE THE DEVICE *//
//* ADDRESS WITH A SLASH (/).  A 3-DIGIT SPECIFIC DEVICE ADDRES*//
//* MAY ALSO BE PRECEDED WITH A SLASH.                         *//
//*                                                            *//
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
//TMAP     EXEC PGM=CTSTMAP,
//  PARM='MAP TEST01,TEST02,TEST03,TEST04,TEST05,TEST06 UNIT=3490'
//* PARM='MAP TEST01 UNIT=CART'
//* PARM='MAP TEST01 UNIT=/2E80'
//*        STEPLIB REQUIRED IF CAI.CTAPLINK NOT IN LINKLST     */
//STEPLIB  DD DISP=SHR,DSN=CAI.CTAPLINK       /* CAI.CTAPLINK   */
//SYSUDUMP DD SYSOUT=*                        /* OPTIONAL       */
//
```

CTSTMAP Output

```
2010/350    10:54:43      TAPEMAP : TAPE CONTENTS IDENTIFICATION UTILITY      PTAINS     0 FILES *** AP COMPLEGE   1

                   TAPE VOLUME : 001519                            OWNER DATA :

FILE       FILE                REC  REC  BLOCK  VOL  BASE    EXPIRE     CREATE    CREATE   CREATE  PSWD        BLOCK

 SEQ    IDENTIFIER     GEN/VER FMT  SIZE  SIZE  SEQ VOLUME    DATE       DATE      JOB      STEP   TYPE   DEN  COUNT

---- ----------------- ------- --- ----- ----- ---- ------ ---------- ---------- -------- -------- ----- ---- ------

   1 BNI01.TAPE4.FILE1         FR   2346 23460    1 001519 ZEROS      2010/344   NOBNI01B STEP1    NONE  CART      2

CA$F068I TAPE MAP COMPLETE : VOLUME CONTAINS     1 FILES ***

CA$F068I TAPE MATE : VOLUME CON
```

Chapter 10: Maintaining the Volume Master File

This section contains the following topics:

Understanding VMF Chaining

CA TLMS uses a highly sophisticated chaining technique to link volume base records that are created when multivolume data sets are created. Chains are broken and returned under normal operating conditions according to the value specified for the BRKCHN system option (see BRKCHN= Break Volume/Data Set Chains (see page 41)). The option can be specified as one of the following:

CLOSE

The chain is broken when a CLOSE is performed on one of the tapes. Specifying CLOSE should only be considered for special processing.

Note: This option is supplied for compatibility with earlier versions and must only be used in special circumstances. Note that chains will still be broken at OPEN, regardless of this option being specified.

OPEN

A chain is broken when one of the volumes in the original chain is opened for output. Each volume within the chain becomes a single volume and all chaining fields (with the exception of tape history) in the associated VMF records are reset. This is the default.

SCRATCH

All chains are broken when the tape is scratched, and previous information (with the exception of tape history) is reset. If a tape is scratched in error, the chain cannot be recreated by unscratching.

The volume chaining verification procedure (CATVCVS) checks the validity of the links between volume base records. This procedure should be executed on a daily basis. (The CATVCVS procedure is detailed in this chapter under "Verifying Volume Master File Chains" on Verifying Volume Master File Chains.)

Chaining Errors

The following conditions are probable causes of chaining errors.

Missing operating system intercepts

Missing OPEN and CLOSE operating system intercepts can cause chaining errors that are compounded if the condition is not corrected. A missing operating system intercept allows multiple chaining on the same volume.

Non-CA TLMS controlled volume in a CA TLMS controlled chain

A chaining pointer could not be established because a non-CA TLMS controlled volume was used in a multivolume chain output.

System or CA TLMS catastrophic failure

A hard z/OS system failure or abend affecting CA TLMS would not allow proper closing of a data set chain.

A tape volume is currently being used and has not gone through CLOSE processing

The tape is open for output and CLOSE information has not been posted. When examining the CATVCVS report, ensure that the tape is not currently being used.

TLMSII.DUMMY.DSN is a chained data set

CA TLMS re-chains when a multivolume or multi-data set is read as input. This is not a true chaining error.

Note: For information on the INPUT system option, see the section INPUT System Option—Input Data Set Processing Requirements (see page 53).

Application is not processing normal IBM OPEN and CLOSE logic

If certain application data sets are continually listed on the CATVCVS report, they may be bypassing standard IBM OPEN and CLOSE logic. Contact CA for assistance. See the chapter in this guide titled "Troubleshooting" for further instructions.

Verifying Volume Master File Chains

CATVCVS is a maintenance utility procedure that detects invalid internal chaining of a multivolume and multi-data set, or errors that exist in the free chain of unused AUX records. This PROC should only be used by the systems programmer responsible for maintaining CA TLMS.

CATVCVS should be executed on a regular basis; explanations and corrective actions should be taken for every error message issued. Please note that open-ended chains will occur if:

- This PROC is executed during tape processing
- The manual chaining commands are used incorrectly
- A catastrophic system failure occurs during tape processing

Process

The only input required for CATVCVS is the CA TLMS Volume Master File.

CATVCVS Procedure JCL

The following JCL is used to verify and report on Volume Master File chains:

```
//*****************************************************************
//*             ****  PROCNAME=CATVCVS  ****                      *
//*****************************************************************
//**             PROCEDURE TO VERIFY VOLUME CHAINS               *
//*****************************************************************
//*
//CATVCVS  PROC A='*',                          SYSOUT=A
//             BUFNO=10,                        BUFNO
//             LOAD='CAI.CTAPLINK',               CA LOADLIB
//             OPTS='CAI.CTAPOPTN(TLMSIPO)',    TLMS OPTIONS
//             VMF='CAI.TLMS.VMF'               VOLUME MASTER FILE
//*
//TLMSVCVS EXEC PGM=TLMSVCVS
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR
//*
//SYSPRINT DD  DCB=BLKSIZE=133,SYSOUT=&A
//CAIVMF   DD  DSN=&VMF,DISP=SHR
//*
//CAIVMFS  DD  DSN=&VMF,DISP=SHR,
//             DCB=(BUFNO=&BUFNO,OPTCD=C)
//*
//SYSUDUMP DD  SYSOUT=&A
//*
```

Starter JCL

```
//TLMSVCVS JOB
//TLMSVCVS EXEC CATVCVS
//
```

After errors have been indicated for specific volumes, closely review the volume and all associated data sets and chained volumes. Corrective action is dependent on the error message produced. The error messages generated by CATVCVS are detailed in the *Message Reference Guide.*

```
CA TLMS NNn  0508TLC60                                                                    mm/dd/yyyy  08.03.00
CAT6203W VSN=C00051 VOLSEQ=001 VOLCNT=000  VOLUME COUNT LESS THAN VOLUME SEQUENCE
CAT6203W VSN=C00052 VOLSEQ=002 VOLCNT=000  VOLUME COUNT LESS THAN VOLUME SEQUENCE
CAT6203W VSN=C00053 VOLSEQ=003 VOLCNT=000  VOLUME COUNT LESS THAN VOLUME SEQUENCE
CAT6203W VSN=C00054 VOLSEQ=004 VOLCNT=000  VOLUME COUNT LESS THAN VOLUME SEQUENCE
CAT6203W VSN=C00055 VOLSEQ=005 VOLCNT=000  VOLUME COUNT LESS THAN VOLUME SEQUENCE
//*****************************************************************
```

Correcting Chaining Errors

Chaining errors are corrected by breaking valid or invalid volume/data set chains, and rebuilding them. Update commands are used to accomplish these functions. These commands can only be used through the online CTS inquiry/update facility, the TSO CLIST CATLTSO or the CATINQR PROC batch report facility. They cannot be used with TLTPISPF online.

Breaking Valid Chains

This process breaks volume chains that have been previously established. The chain must be broken at the base record of a scratch volume. When input tape data sets that were created outside CA TLMS are concatenated together, the volumes are chained in the VMF (unless a system modification specifies not to chain on input). To break a volume/data set chain, use the following command:

```
UPV vvvvvv,CHV=BRKCHN
```

Where:

vvvvvv

Represents the base volume serial number of the multivolume data set or multi-data set volume.

This command resets the volume sequence, volume count, file sequence and file count fields on the VMF. It also clears all data set related fields for all volumes in the chain.

The following steps are used to reestablish chains after they are broken:

1. Establish the base volume and update the data set-related fields (this reinitializes the VOLSEQ, VOLCNT, FILESEQ, and FILECNT fields).
2. Use the appropriate command to rebuild the volume/data set chain (see "Rebuilding Volume/Data Set Chains" on Rebuilding Volume/Data Set Chains).

Breaking Invalid Chains

If the chain you are breaking is **invalid**, you must issue the UPV command for *each* volume of the chain in the following format:

```
UPV vvvvvv,EXPDT=60001
```

Where:

vvvvvv

Represents the volume serial number.

If an invalid chain is established, but the data set statistics are correct, then only the chaining information needs to be corrected. This command resets all volume/data set chain pointers and initializes the volume sequence, volume count, file sequence and file count to one, but leaves all data set information intact for the first data set (only) on the volume.

Important! Exercise extreme caution when using this command. If all volume/data set chains are not properly cleared and reset, illogical chaining conditions may result which could cause unpredictable results. Be advised to consult with CA CA TLMS support personnel before using this command.

Example

If processing is bypassed for a volume that is part of a chain and that volume is used as input or concatenated with other volumes, it becomes a volume in two chains. This is an example of an incorrectly established chain. To resolve this chaining error, use the UPV vvvvvv,EXPDT=60001 command to break the chain; issue the command for *each* volume belonging to the invalid chain.

Clearing Volume/Data Set Related Fields

This process clears all data set related and volume related fields in the Volume Master File. The associated volume must be in scratch status, and cannot be chained.

The following command is used to clear all data set related and volume related fields:

```
UPV vvvvvv,CLEAR
```

Where:

vvvvvv

Is the volume serial number of the base record.

Volume history and volume maintenance are not cleared by this command.

Rebuilding Volume/Data Set Chains

If Volume Master File updates are lost through either destruction of the transaction log or failure to properly back up the CA TLMS VMF, you may need to rebuild chains for multivolume and/or multi-data sets. If this becomes necessary, you must reconstruct as much information about the volumes and data sets as available before attempting to rebuild the chains.

Either CATLTSO or the CATINQR procedure can be used to rebuild volume/data set chains.

While data set chaining and volume chaining are very similar, each process is discussed separately on the following pages. In both cases, however, the rules for adding a new member to a chain are the same:

- You must always address the base volume of a multivolume data set or multivolume/multi-data set.
- You must always add the next logical volume or data set in the series. That is, if you wish to add 3 more volumes to an existing chain of 4 volumes, you must first add VOLSEQ=5, then VOLSEQ=6, then VOLSEQ=7. Any attempt to enter VOLSEQ=7 out of sequence will be rejected.

If both data set and volume chains have to be rebuilt, process data set chaining first.

Data Set Chaining

To chain a data set to a given volume or set of volumes, the following command is used:

```
UPD vvvvvv,CHD=dsn,FILESEQ=nnnn
```

Where:

vvvvvv

Represents the base volume serial number for the multi-data set.

dsn

Specifies the data set name of the data set to be chained (up to 44 characters).

nnnn

specifies the relative file number of the data set being chained (this number must be one greater than the last data set currently in the chain).

Important! Do not use FILESEQ=001 for the first data set in the chain. This field must be used starting with the second data set (FILESEQ=002).

Once the data set has been successfully chained, you may use the UPD command to update the other data set related fields, if necessary.

Example

The following is an example of three data sets on a single volume that needs re-chaining. The volume serial number is 000001. The data set on file one is USERDSN.FILE1, the second is USERDSN.FILE2, and the third is USERDSN.FILE3.

```
UPD 000001,DSN=USERDSN.FILE1              -ADDRESSES BASE VOLUME
UPD 000001,CHD=USERDSN.FILE2,FILESEQ=002 -CHAINS FILE2 TO BASE DSN
UPD 000001,CHD=USERDSN.FILE3,FILESEQ=003 -CHAINS FILE3 TO BASE DSN
```

Volume Chaining

To chain a volume, the following command is used:

```
UPV vvvvvv,CHV=xxxxxx,VOLSEQ=nnn
```

Where:

vvvvvv

Represents the base volume serial number of the multivolume data set.

xxxxxx

Represents the volume serial number of the volume to be chained into the set.

nnn

Represents the volume sequence number of the volume to be chained. This value must be 1 greater than the last volume sequence number currently in the chain.

Once all the required volumes have been chained, UPV command may be used to update other volume related fields, if necessary.

Examples

The following is an example of multivolume/multi-data set chaining.

```
UPD XXX001,DSN=CHAIN1
UPD XXX001,CHD=CHAIN2,FILESEQ=002
UPD XXX001,CHD=CHAIN3,FILESEQ=003
UPD XXX001,CHD=CHAIN4,FILESEQ=004
UPV XXX001,CHV=XXX002,VOLSEQ=002
UPD XXX002,DSN=CHAIN4,FILESEQ=004
UPD XXX002,CHD=CHAIN5,FILESEQ=005
UPD XXX002,CHD=CHAIN6,FILESEQ=006
UPV XXX001,CHV=XXX003,VOLSEQ=003
UPD XXX003,DSN=CHAIN6,FILESEQ=006
UPD XXX003,CHD=CHAIN7,FILESEQ=007
```

The following is an example of a three-volume data set that requires re-chaining. The data set is USERDSN.BACKUP. Volume serial numbers are 000001, 000002 and 000003.

```
UPD 000001,DSN=USERDSN.BACKUP    <- ADDRESSES BASE VOLUME
UPV 000001,CHV=000002,VOLSEQ=002 <- CHAINS VOLSER 000002 TO BASE V
OLUME
UPD 000002,DSN=USERDSN.BACKUP    <- UPDATES VOLSER 000002 DATA SET
 NAME
UPV 000001,CHV=000003,VOLSEQ=003 <- CHAINS VOLSER 000003 TO BASE V
OLUME
UPD 000003,DSN=USERDSN.BACKUP    <- UPDATES VOLSER 000003 DATA SET
 NAME
```

The following is an example of seven data sets on three volumes that need re-chaining. The volume serial numbers are 000001, 000002 and 000003.

Volser 000001 has the following data sets:

```
FILE ONE    - USERDSN.FILE1
FILE TWO    - USERDSN.FILE2
FILE THREE  - USERDSN.FILE3
FILE FOUR   - USERDSN.FILE4
```

Volser 000002 has the following data sets:

```
FILE FOUR   - USERDSN.FILE4   <-DATA SET SPANS FROM 000001 onto 00
0002
FILE FIVE   - USERDSN.FILE5
FILE SIX    - USERDSN.FILE6
```

Volser 000003 has the following data sets:

```
FILE SIX   - USERDSN.FILE6  <--DATA SET SPANS FROM 000002 onto 000003
FILE SEVEN  - USERDSN.FILE7
```

Example of multi-data set chaining:

```
UPD 000001,DSN=USERDSN.FILE1             <- ADDRESSES BASE DATA SE
T
UPD 000001,CHD=USERDSN.FILE2,FILESEQ=002 <- CHAINS FILE2 TO BASE D
SN
UPD 000001,CHD=USERDSN.FILE3,FILESEQ=003 <- CHAINS FILE3 TO BASE D
SN
UPD 000001,CHD=USERDSN.FILE4,FILESEQ=004 <- CHAINS FILE4 TO BASE D
SN
UPV 000001,CHV=000002,VOLSEQ=002         <- CHAINS VOLSER 000002 T
O BASE
UPD 000002,DSN=USERDSN.FILE4,FILESEQ=004 <- CHAINS FILE4 TO 000002
UPD 000002,CHD=USERDSN.FILE5,FILESEQ=005 <- CHAINS FILE5 TO 000002
UPD 000002,CHD=USERDSN.FILE6,FILESEQ=006 <- CHAINS FILE6 TO 000002
UPV 000001,CHV=000003,VOLSEQ=003         <- CHAINS VOLSER 000003 T
O BASE
UPD 000003,DSN=USERDSN.FILE6,FILESEQ=006 <- CHAINS FILE6 TO 000003
UPD 000002,CHD=USERDSN.FILE7,FILESEQ=007 <- CHAINS FILE7 TO 000003
```

Taking a Snapshot of VMF Records

The CATSNAP procedure is provided to aid in the diagnosis of CA TLMS internal processing problems. It issues the IBM SNAP macro for each volume serial number supplied in the input stream. The output snapshot can then be used in detailed analysis of logic flow within the system. Multi-data set records (if applicable) are automatically snapped for the first volume in a chain of volumes.

This procedure is frequently used in conjunction with the CATVCVS or CATVMFV procedures.

Process

The input file to the CATSNAP procedure is an 80-byte statement, where the first six characters identify the volumes serial number to be diagnosed. Any number of input statements can be supplied.

CATSNAP Procedure JCL

The following procedure is used to produce a snapshot of the VMF record.

```
//*************************************************************
//*             ****  PROCNAME=CATSNAP  ****                  *
//*************************************************************
//**             PROCEDURE TO SNAPSHOT THE CA TLMS VMF         *
//*************************************************************
//*
//CATSNAP  PROC A='*',                          SYSOUT CLASS
//             LOAD='CAI.CTAPLINK',               CA LOAD LIB
//             OPTS='CAI.CTAPOPTN(TLMSIPO)', TLMS OPTIONS
//             VMF='CAI.TLMS.VMF'            VOLUME MASTER FILE
//*
//TLMSSNAP EXEC PGM=TLMSSNAP
//*
//STEPLIB  DD  DSN=&LOAD,DISP=SHR
//*
//TLMSOPTS DD  DSN=&OPTS,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=&A,DCB=BLKSIZE=133
//*
//SNAPSHOT DD  SYSOUT=&A
//*
//CAIVMFI  DD  DSN=&VMF,DISP=SHR
//*
//SYSUDUMP DD  SYSOUT=&A
//*
//*************************************************************
```

Starter JCL

```
//TLMSNAP  JOB
//TLMSSNAP EXEC CATSNAP
//SYSIN    DD   *
  <VSNs in columns 1-6>
//
```

CATSNAP Output

```
YOUR TLMS NN.n COMPANY NAME             VOLUME MASTER FILE SNAP UTILITY
CA TLMS NN.n yymmTLrrr

SNAPID=001  000041
SNAPID=002  000010
SNAPID=     TAPE06   NOT ON VMF
JOB JOCRASNP         STEP TLMSSNAP        TIME 165428   DATE yyddd    ID = 001
PSW AT ENTRY TO SNAP   078D1000 00006EFE        ILC 2   INTC 0033
-STORAGE
000078E0          C240F0F0 F0F0F4F1 000C000C    00000000 00000000 00000000 00000
00007900 00000000 00000000 40404040 40400000    40404040 40400000 40404040 40400
00007920 40404040 40400000 40404040 40400000    09600000 000CC1C1 40404040 4040E
00007940 40404040 0000000C 00000001 00010001    00000000 00000000 00000000 000C0
00007960 000CF540 E2F24040 40404040 4040D4C5    40404040 40400000 C4C34040 40400
00007980 000C0000 000C0000 000C0000 000CF040    F0404000 00F04040 0000F040 40000
000079A0 40400000 F0404000 00F04040 00000000    00000000 00000000 00000000 00000
000079C0 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000
000079E0 00000000 00000000 00000000 F0F0F0F0    F4F10000 40404040 40404040 40404
00007A00 40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404
00007A20 00404040 00000000 00000000 00000000    0000000C 0000000C 0000000C 00000
00007A40 40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404
00007A60 40404040 40404040 40404040 40404040    40404040 40400000 000C4040 40404
00007A80 40404040 40404040 40404040 00000000    00000000 00000000 00000000 00000
00007AA0 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000
```

```
JOB JOCRASNP         STEP TLMSSNAP        TIME 165428   DATE yyddd    ID = 001

                  DUMP INDEX
                  ----------
 DATA AREAS                          PAGE NUMBER
-----------                          -----------
 STORAGE AREAS..... 00000001
END OF DUMP
```

Correcting WORM Field Errors in VMF Records

When a Write Once Read Many (WORM) volume is opened for input or output, CA TLMS will verify that the "World Wide ID" (WWID) and "Write Mount Count" (WMC) values captured from the IBM 3592 hardware match those values recorded in the VMF. When a mismatch occurs, CA TLMS rejects the tape and issues a message containing the hardware WWID and WMC.

Since WORM volumes are normally reserved for sensitive data, the proper authorities in your company should be contacted before any corrective action is taken. If it is decided to reset the VMF to accept the existing WORM values, the following procedure may be taken.

Resetting WORM values

This command requires MAINTENANCE level security. It will clear the WWID and WMC fields in the VMF to binary zeros. This will allow CA TLMS to accept the hardware values the next time the volume is accessed. Only the volser is verified.

```
UPV vvvvvv,CLRWRM
```

where:

vvvvvv

Represents the volume serial number of the WORM volume to reset.

Chapter 11: Security System Interface

This section contains the following topics:

About Security

CA TLMS provides protection for tape processing through the security system interface. This interface is designed to function with CA ACF2 Security, CA Top Secret Security, IBM's RACF or any other security system that is IBM SAF compatible. CA TLMS is not security product-sensitive, but it is determined through the interface which system security product is to be called.

Tape processing protection is completely controlled by the user with the optional selection of the security desired, using the CA TLMS system options in member TLMSIPO in CAI.CTAPOPTN. These options indicate when system security is to be called and what functions are to be controlled through CA TLMS.

Selectable security functions include data set name access during tape OPEN and CLOSE, label protection for BLP, NL, NSL and foreign volume processing (LABEL=EXPDT=98000), online access protection using online panel (TLTPISPF only) and command control, and batch job protection by checking access to a data set and to the command groups.

User exit TLMSXSEC is provided to allow user interface with CA TLMS and the system security component. Each call to security is made available to the user exit to view or alter, as required by the installation. (See Chapter 3, *User Exits and Macros*, for detailed information on the use of TLMSXSEC.)

Security Interface Options

CA TLMS provides security interface options that control external security processing. Each option can be activated/deactivated individually using the associated parameters in member TLMSIPO of CAI.CTAPOPTN. Instructions for setting these options are contained in Chapter 3.

CA TLMS also furnishes several security interface options for each of the major operating system intercept points within the system. These intercept points are OPEN/CLOSE/EOV, online access and control functions, such as tape label processing, online panel access, and online and batch command authorization.

Global System Security Options

All security interface intercept points are activated and deactivated by the CA TLMS global security option, SECURE=. If this option is set to NO, all other security option values are ignored. That is, all the security intercept points will be turned off within CA TLMS, including the user exit TLMSXSEC. If set to YES, each of the other security option values are activated or deactivated, depending on their respective values.

Online processing can be globally controlled with the option, INQACC=. If this option is set to NO, security processing will be deactivated for online. This applies to the full screen online processor, TLTPISPF only. The other "online" processors, such as TLMSTSO, are command processors and are controlled under the control functions intercept for commands.

OPEN/CLOSE/EOV Intercept Options

There are two CA TLMS security options available at this intercept point; SECOPN= and SECCLS=. Both of these options are processed by the CA TLMS SVC at data set/volume OPEN and CLOSE.

SECOPN=

Tells CA TLMS at data set OPEN to verify that the tape user is authorized to read or write the data set to be accessed. With this option set to YES, the IDSNVER= option is required to be set to YES, because both OPEN INPUT and OPEN OUTPUT data set checking are performed. CA Top Secret Security users can use this option in place of the OPEN user exit code that performed this same function in TLMSXOPN.

The information passed to the TLMSXSEC user exit and the system security component are:

```
CLASS=DATASET
ENTITY=data.set.name
VOLSER=volser
ACCESS=CREATE (NEW)
       UPDATE (MOD)
       UPDATE (OLD) OPENED FOR OUTPUT
       READ   (OLD,SHR) OPENED FOR INPUT
```

SECCLS=

Tells CA TLMS to check data set DISP= at CLOSE (second DISP) and job abend (third DISP). The accessing data set user is checked for the ability to delete a file, as would be the case in normal processing for this JCL DISP: DISP=(OLD,DELETE). The user may have been authorized for read (first DISP), but not for scratch (second DISP). The third DISP value, if coded, is checked in the same way if the job or task abends. CA TLMS will not allow the data set to be deleted from the VMF when there are any conflicts in the user's authority and the data set being accessed.

The information passed to the TLMSXSEC user exit and the system security component are:

```
CLASS=DATASET
ENTITY=data.set.name
VOLSER=volser
ACCESS=SCRATCH (,DELETE) or (,DELETE,DELETE) JCL DISP
```

Online Intercept Options

CA TLMS provides three security options at this intercept point; INQACC= (which is the global online security option previously explained), SECINQ= and PROMPT=. These options only control the full screen online processor, TLTPISPF. The TLTPISPF security table, TLMSUTAB, remains in use, but performs a different function if online security has been activated.

SECINQ=

Tells CA TLMS that a data set security check is to be performed when attempting to display detailed data set information. This means a requesting user must at least have "read" access to a data set before its information can be viewed. If this option is set to NO, inquiries will be allowed without checking the user's ability to access the requested data set.

The information passed to the TLMSXSEC user exit and the system security component are:

```
CLASS=DATASET
ENTITY=data.set.name
VOLSER=volser
ACCESS=READ   (SHR) AS IF OPENED FOR INPUT
```

PROMPT=

If set to NO, tells CA TLMS to bypass the security logon panel when logging on to TLTPISPF and obtain the logon user ID from the operating system. This option is always forced to NO when security is activated for online. If security is inactive for online processing, PROMPT= can be set to YES or NO. If set to YES, TLTPISPF will display the security logon panel (SC01), where the user must enter both user ID and password.

In either case, if online security is deactivated, the TLTPISPF user ID and password table, TLMSUTAB, must contain the logon user ID and the level of access, inquiry or update. If PROMPT=YES is specified, TLMSUTAB must also contain the user's password.

TLMSUTAB (TLTPISPF Table) Options

Use the TLTPISPF table TLMSUTAB to control online access to data set information. When security is deactivated for TLTPISPF, this table defines who can logon to TLTPISPF and the functions they can to perform.

When security is activated for online (INQACC=YES), this table is used to define a user and the level of access allowed without invoking a system security data set access check.

Example: TLMSUTAB Table

In this example, a tape librarian needs to update a data set panel (DS02) with SCRATCH=YES but does not have update access to the data set being scratched. To avoid having to set up system security to allow for that type of access, the TLMSUTAB entry is coded as follows:

```
USER="LIBUSEID",PWD=,ACCESS=U
```

USER

Specifies the librarians TSO user ID.

PWD

Is null, because this is verified at TSO logon time.

ACCESS

Set to UPDATE status.

Note: TLMSUTAB replaces TLTPOPTS which was in previous releases.

Control Functions

CA TLMS security intercepts for control functions include label access control, which occurs at data set/volume OPEN. Protection is provided by security options BLPSEC=, NLSEC=, NSLSEC= and FORSEC=. Additional control functions protect panel access for TLTPISPF and command control used by TLTPISPF, batch processing, and TLMSTSO.

Label Control Function

Label control consists of four security options; BLPSEC= for controlling the ability to process BLP tapes, NLSEC= and NSLSEC= for controlling No Label (NL) and Nonstandard Label (NSL), respectively, and FORSEC= to control use of foreign volumes.

Any of these four security options may be specified as YES or NO. This activates/deactivates the security intercepts in CA TLMS to perform a security check to see if the caller has access to any of these label types. Calls made to the security interface will indicate whether the label access is for a volume within or outside the range of the VMF. That is, the volume serial is resident or not resident on the VMF.

The information passed to the TLMSXSEC user exit and the system security component are:

```
CLASS=CATAPE      (CLASS must be defined to system security.)
ENTITY=xxxRES     Volume is resident on the VMF.
       xxxNORES   Volume is not resident on the VMF.
                     xxx is BLP, NL, NSL or FOR
ACCESS=READ       Volume opened for input
       UPDATE     Volume opened for output
```

Panel Control for TLTPISPF

CA TLMS automatically turns on the security intercept for panel control in TLTPISPF when the online global security option, INQACC= is specified as YES. Panel control allows TLTPISPF to restrict access to certain inquiry and update panels that can be displayed online.

User access to a display panel is either ON or OFF, meaning the user either has access to a panel or does not. Panel access only controls the ability to view a panel. Once authorized to access a panel, the user still must have authority to view or update the information on that panel. Inquiry and update authority from an authorized panel is handled through command processing control.

The panel "entities" to which access is controlled through TLTPISPF and the system security component are:

```
AU01, AUXILIARY MESSAGES : TLPAU01
DL01, VOLUME SERIAL LIST : TLPDL01
DS01, DATA SET SELECTION : TLPDS01
DS02, DATA SET DETAIL    : TLPDS02
CT01, DATA SET CATALOG   : TLPCT01
MS01, MISC. SELECTION    : TLPMS01
OP01, OPTIONS SELECTION  : TLPOP01
VL01, VOLUME SELECTION   : TLPVL01
```

The information passed to the TLMSXSEC user exit and the system security component are:

```
CLASS=PANEL       (CLASS must be defined to system security)
ENTITY=TLP****    Panel name definition (**** is one or more of the panel
                  names listed above)
ACCESS=READ (YES) Panel access is granted
       NONE (NO)  Panel access is restricted
```

Command Processing Control

Several program elements inside and outside of CA TLMS use direct command processing to the CA TLMS address space (CTS). These commands can range from displaying date format information to breaking data set and volume chains on the VMF. When security is activated for CA TLMS, command control intercepts are automatically set to allow access checking with the system security interface.

Note: For information on commands and their syntax, see the *User Guide*.

It is important to understand that command authorization is performed on the user ID, not by program name. For example, TLMSTRS can issue an UPDATE VOLUME SCRATCH DATA SET (UPV 123456, SCRATCH=YES,DSN=(CAI.EXPIRED.TAPE) the same as a user entering commands in TLMSTSO command processor. In both cases, the user running the batch job or entering online commands, once authorized, can issue or cause commands to be issued from any CA TLMS facility.

When defining security rules for those users that will be allowed to issue commands directly and indirectly, ensure that program interfaces supplied for use with CA TLMS are also authorized for any commands they may issue, including the CA TLMS address space.

For security purposes, and to ease the security rule definition process, CA TLMS commands are broken down into the four entity groups *maintenance, librarian, update* and *read*. Although each command can be defined as a single entity, the following is a summary of the four command groups. The command entities to which access is controlled through TLMSCMND and the system security component are:

Maintenance Commands

TLMcmd, where *cmd* is one of the following:

BRK

Update volser (CHV=BRKCHN)

CLR

Update volser (CLEAR)

CHN

Update volser (CHD=dsname,CHV=volser)

CLW

Update volser (CLRWRM)

Librarian Commands

TLLcmd,where *cmd* is one of the following:

AMA

Add message

AMD

Delete message

CLN

Clean

CER

Certify

SRV

Service in or out by volume

SCR

Update volser (SCRATCH=YES/NO)

CDS

Update volser (CDS=nnn)

OWN

Update volser (OWNER=name)

RTN

Update the retention schedule

Bypass DSN check:

TLUVMFx, where *xcmd* is one of the following:

U

Bypass DSN check when user access is Update

R

Bypass DSN check when user access is Inquiry

Update Commands

TLUcmd, where *cmd* is one of the following:

UPV

Update volume

UPD

Update data set

Read (Display) Commands

TLRDV*, where * is one of the following:

D

Display volume status, data set characteristics

H

Display volume status and history

M

Display volume status and maintenance

R

Display volume status and retention

L

Display volume list

A

Display all volume information

TLRD*, where * is one of the following:

N

Display volume information by DSN

C

Display data set information for cataloged data set

M

Display the text of an auxiliary message.

The information passed to the TLMSXSEC user exit and the security system component are:

```
CLASS=CACMD      (CLASS must be defined to system security)
ENTITY=TLM***    Maintenance commands
       TLL***    Librarian commands
       TLVMFU    Bypass DSN security checks for update access
       TLVMFR    Bypass DSN security checks for inquiry access
       TLU***    Update commands not covered by TLM or TLL
       TLR***    Read or inquiry commands (*** is specified as
                 listed above)
ACCESS=YES       User has access
       NO        User does not have access
```

CA ACF2 Security Setup

The CA TLMS external security system does *not* require the SAF option of CA ACF2 Security. Only two setup steps are required:

1. Define the resource types.
2. Define the access rules.

Note: Prior to activating the CA TLMS external security system, CA ACF2 Security Version 5.2, Level 9010 or higher must be installed. Additionally, PTF TW86655 must be verified as applied.

Defining Resource Types

Three new types are needed for CA TLMS external security:

- CAT (for CATAPE resources)
- CAC (for CACMD resources)
- PAN (for PANEL resources)

The following CA ACF2 Security commands add these types to the resident directory (if they are not already defined to the resident directory):

```
SET CONTROL(GSO)
CHANGE INFODIR TYPES(D-RCAT,D-RCAC,D-RPAN)
```

You must then issue the refresh command for INFODIR:

```
F ACF2,REFRESH(INFODIR)
```

If Resource Type is altered, a rebuild of associated directory is required.

```
F ACF2,REBUILD(DIRECTORY TYPE CODE)
```

Defining Access Rules

Once the Resident Directory has been refreshed, you may define the access rules. The following are samples of the commands used:

```
$KEY(TLM***) TYPE(CAC)
UID(xxxxxxxx) ALLOW
```

Repeat the above commands as required, specifying $KEY for each of the following. If there are individual commands that are to be controlled, replace asterisks (***) with the command name.

(TLM*)**

Maintenance command access

(TLL*)**

Librarian command access

(TLU*)**

Update command access

(TLR*)**

Read or Inquiry command access

(TLVMFU)

Bypass DSN check for Update access

(TLVMFR)

Bypass DSN check for Inquiry access

Below are sample commands for command processing. There are no service levels for command processing; access is either ON or OFF.

```
$KEY(TLR***) TYPE(CAC)              (Allow all users access to
UID(*) ALLOW                         read/inquiry commands.)

$KEY(TLU***) TYPE(CAC)              (Allow all TSO users access to
UID(T-) ALLOW                       update commands.)
```

The following is a list of all resources for the CATAPE type:

$KEY(NLRES)	Label=NL, defined to CA TLMS
$KEY(NLNORES)	Label=NL, not defined to CA TLMS
$KEY(NSLRES)	Label=NSL, defined to CA TLMS
$KEY(NSLNORES)	Label=NSL, not defined to CA TLMS
$KEY(BLPRES)	Label=BLP, defined to CA TLMS
$KEY(BLPNORES)	Label=BLP, not defined to CA TLMS
$KEY(FORRES)	EXPDT=98000, defined to CA TLMS
$KEY(FORNORES)	EXPDT=98000, not defined to CA TLMS

Examples

This is an example of the rule to allow all users read access to NL tapes controlled by CA TLMS:

```
$KEY(NLRES) TYPE(CAT)
UID(*) SERVICE(READ) ALLOW
```

This is an example of the rule to allow all users read access to BLP tapes that are *not* controlled by CA TLMS:

```
$KEY(BLPNORES) TYPE(CAT)
UID(*) SERVICE(READ) ALLOW
```

The following is a list of resources for PANEL type:

```
$KEY(TLP****)
UID(*) SERVICE(READ) ALLOW
```

Examples

This is an example of the rule to allow all users access to the data set detail panel DS02. Access to data select panel (DS01) is required also.

```
$KEY(TLPDS**) TYPE(PAN)
UID(*) SERVICE(READ) ALLOW
```

Usage Notes

CA ACF2 Security philosophy prescribes that all resources are protected by default. Activation of CA TLMS external security requires that proper authorization for resources being checked is established within CA ACF2 Security prior to activation of the CA TLMS external security options. It is not possible to deactivate CA TLMS external security options without changing the TLMSIPO member of CAI.CTAPOPTN, and then either performing a CAIRIM CA TLMS REFRESH or an IPL of the entire operating system. Obviously, a REFRESH is much easier than a system IPL.

CA Top Secret Security Setup

Note: Prior to activating the CA TLMS external security system, CA Top Secret Security Version 4.2, Level 9011 or higher must be installed.

The following steps are required for implementation of the CA TLMS external security system in a CA Top Secret Security environment:

Step 1

Assign ownership of all entities to be protected. The following commands are used to perform this step.

```
TSS ADD(acid) CACMD(TLM)
TSS ADD(acid) CACMD(TLL)
TSS ADD(acid) CACMD(TLU)
TSS ADD(acid) CACMD(TLR)
TSS ADD(acid) CACMD(TLVMFU)
TSS ADD(acid) CACMD(TLVMFR)
TSS ADD(acid) CATAPE(NLRES)
TSS ADD(acid) CATAPE(NLNORES)
TSS ADD(acid) CATAPE(NSLRES)
TSS ADD(acid) CATAPE(NSLNORES)
TSS ADD(acid) CATAPE(BLPRES)
TSS ADD(acid) CATAPE(BLPNORES)
TSS ADD(acid) CATAPE(FORRES)
TSS ADD(acid) CATAPE(FORNORES)
TSS ADD(acid) PANEL(TLP****)
```

Step 2

Permit users access as desired. Use the commands in one of the following syntax diagrams:

```
TSS PERMIT(user1) CATAPE(NLRES)
       ACCESS[NONE|READ,UPDATE|ALL]
```

or

```
TSS PERMIT(user1) CACMD(TLR)
```

or

```
TSS PERMIT(user1) PANEL(TLPD) READ
```

Usage Notes

The control option TAPE should be set to OFF in the CA Top Secret Security parameter file. This will prevent CA Top Secret Security from being invoked by MVS.

CA TLMS security option SECOPN=YES should be used. This eliminates the need to use supplied CA Top Secret Security code for user exit TLMSXOPN.

Activation of CA TLMS external security requires that proper authorization for resources being checked is established within CA Top Secret Security prior to activation of the CA TLMS external security options. It is not possible to deactivate CA TLMS external security options without changing the TLMSIPO member of CAI.CTAPOPTN, and then either performing a CAIRIM CA TLMS REFRESH or an IPL of the entire operating system. Obviously, a REFRESH is much easier than a system IPL.

IBM RACF Security Setup

The following steps are required for implementation of the CA TLMS external security system in an IBM RACF environment. (Each step is discussed in detail in the subsequent paragraphs.)

Step 1

Assemble the Class Descriptor table.

Step 2

Assemble the RACF Router table.

Step 3

Define CA TLMS resources to RACF.

Step 4

Assign authority for CA TLMS users.

Step 5

Activate the CA TLMS classes.

Step 6

Assemble the CAISSF module, CAS9SAFC. (See the *CA Common Services* for *z/OS Installation Guide* for details.)

Class Descriptor Table

The Class Descriptor table is used to describe resource classes to be used by the RACF security system. There are three classes used by CA TLMS: CA@MD, CA@APE and PA@EL.

```
*
*  RACF RESOURCE CLASS DESCRIPTOR TABLE FOR USE WITH CA TLMS
*
         ICHERCDE CLASS=CA@MD,           CA TLMS COMMAND      X
               id=128,                   CHECKING             X
               FIRST=ALPHA,                                   X
               POSIT=25,                                      X
               DFTUACC=NONE
*
         ICHERCDE CLASS=CA@APE,          CA TLMS RESOURCE     X
               id=129,                   CHECKING             X
               FIRST=ALPHA,                                   X
               OTHER=ANY,                                     X
               POSIT=26,                                      X
               DFTUACC=NONE
*
         ICHERCDE CLASS=PA@EL,           CA TLMS TLTPISPF     X
               id=130,                   PANEL CHECKING       X
               FIRST=ALPHA,                                   X
               OTHER=ANY,                                     X
               POSIT=27,                                      X
               DFTUACC=NONE
*
```

Usage Notes

Modification to the Class Descriptor table should be done by the person responsible for installation and maintenance of RACF. An IPL of the operating system is required for this module to become active.

The values of ID and POSIT may have to be changed, as required, at your installation. Consult your IBM RACF manual for more information.

The class names may be changed based on the Class Descriptor table in the CAS9SAFC module supplied with CA TLMS. See the *CA Common Services for* z/OS *Installation Guide* for details on the CAS9SAFC module.

RACF Router Table

The RACF Router table associates z/OS Router invocations with RACF functions. Three entries are required for use by CA TLMS:

```
*
*  RACF ROUTER TABLE FOR USE WITH CA TLMS
*
         ICHRFRTB CLASS=CA@MD,           CA TLMS COMMAND      X
               ACTION=RACF               PROCESSING
*
         ICHRFRTB CLASS=CA@APE,          CA TLMS RESOURCE     X
               ACTION=RACF               PROCESSING
*
         ICHRFRTB CLASS=PA@EL,           CA TLMS PANEL        X
               ACTION=RACF               PROCESSING
*
```

Usage Notes

Modification to the RACF Router table should be done by the person responsible for installation and maintenance of RACF. An IPL is required for this module to become active.

The class names may be changed based on the Class Descriptor table in the CAS9SAFC module supplied with CA TLMS. See the *CA Common Services for* z/OS *Installation Guide* for details on the CAS9SAFC module.

Defining CA TLMS Resources to RACF

The RACF RDEFINE command may be used to define to RACF all resources belonging to the new classes specified in the Class Descriptor table. The following TSO RACF commands may be entered by your RACF security administrator.

```
RDEFINE CA@MD   (TLM**)    UACC(NONE)
RDEFINE CA@MD   (TLL**)    UACC(NONE)
RDEFINE CA@MD   (TLU**)    UACC(NONE)
RDEFINE CA@MD   (TLR**)    UACC(NONE)
RDEFINE CA@MD   (TLVMFU)   UACC(NONE)
RDEFINE CA@MD   (TLVMFR)   UACC(NONE)
RDEFINE CA@APE  (NLRES)    UACC(NONE)  See Notes
RDEFINE CA@APE  (NLNORES)  UACC(NONE)
RDEFINE CA@APE  (NSLRES)   UACC(NONE)  See Notes
RDEFINE CA@APE  (NSLNORES) UACC(NONE)
RDEFINE CA@APE  (BLPRES)   UACC(NONE)
RDEFINE CA@APE  (BLPNORES) UACC(NONE)
RDEFINE CA@APE  (FORRES)   UACC(NONE)
RDEFINE CA@APE  (FORNORES) UACC(NONE)  See Notes
RDEFINE PA@EL   (TLP**)    UACC(NONE)  See Notes
```

Usage Notes

UACC (READ) may be desired for any of the listed resources. Particular attention should be paid to those noted above.

The class names may be changed based on the Class Descriptor table in the CAS9SAFC. See the *CA Common Services for z/OS Installation Guide* for details on the CAS9SAFC module.

Assigning Authority for CA TLMS Users

The RACF PERMIT command is used to define to RACF which users are allowed to use the various controlled functions. This command may be entered either by the RACF security administrator or a person delegated by the administrator and given the appropriate authority. Some examples of the PERMIT command are:

To allow users to read or create NL tapes that are controlled by CA TLMS:

```
PERMIT NLRES     CLASS(CA@APE)  ACCESS(UPDATE) ID(user1,user2,...)
```

To allow users to read BLP tapes that are *not* controlled by CA TLMS:

```
PERMIT BLPNORES  CLASS(CA@APE)  ACCESS(READ)   ID(user1,user2,...)
```

To allow users to use 98000 processing on input or output for tapes that are *not* controlled by CA TLMS

```
PERMIT FORNORES  CLASS(CA@APE)  ACCESS(UPDATE) ID(user1,user2,...)
```

To allow users to add auxiliary messages that do not have librarian command authority:

```
PERMIT TLLAMA CLASS(CA@MD) ACCESS(READ) ID(user1,user2,...)
```

To allow users to access the data set detail panel (DS02) under TSO:

```
PERMIT TLPDS02  CLASS(PA@EL)  ACCESS(READ) ID(user1,user2,...)
```

Usage Notes

The ACCESS value of READ should be specified for all cases except for data set name protection.

The class names may be changed based on the Class Descriptor table in the CAS9SAFC. See the *CA Common Services for z/OS Installation Guide* for details on the CAS9SAFC module.

Activation of CA TLMS external security requires that proper authorization for resources being checked is established within RACF prior to activation of the CA TLMS external security options. It is not possible to deactivate CA TLMS external security options without changing the TLMSIPO member of CAI.CTAPOPTN, and then either performing a CAIRIM CA TLMS REFRESH or an IPL of the entire operating system. Obviously, a REFRESH is much easier than a system IPL.

Activating the CA TLMS Classes

The RACF SETROPTS command is used to activate CA TLMS security classes which were added to the Class Descriptor table. This command must be entered by the RACF Security Administrator. The following is an example of the SETROPTS command:

```
SETROPTS CLASSACT(CA@MD,CA@APE,PA@EL)
SETROPTS NOCLASSACT(TAPEVOL)
```

Usage Notes

Other options may be added to the SETROPTS command, such as those related to statistics gathering and auditing, if desired.

For the following reasons, CA recommends that TAPEVOL not be used:

- CA TLMS cannot validate a user's authorization to access a volume.
- The deletion of a discrete profile for volumes scratched by TLMSRACF is done in the inverse order that tapes are created.

Because RACF Version 1.7 provides for tape data set name access authorization verification, use of the CA TLMS security option SECOPN=YES would duplicate processing that RACF has already performed. (See the *RACF Security Administrator's Guide, TAPEDSN Option*, for information on implementing this facility.) It is recommended that both CA TLMS and RACF TAPEDSN be used. RACF verification occurs after CA TLMS validation and updating of the VMF. If RACF disallows the request (and SECOPN=YES was not used), the VMF is out of synchronization with the header label on the tape, causing a NOT SCRATCH condition when the tape is subsequently used for output.

Assembling CAS9SAFC

Module CAS9SAFC must be assembled using the correct system macro libraries active in your system. (See the *CA Common Services for z/OS Installation Guide* for more details.) This module contains a translation table to translate class names. Without modification, CATAPE is translated to CA@APE, CACMD is translated to CA@MD, and PANEL is translated to PA@EL. This translation allows the RACF Class Descriptor table and RACF Router table to be assembled without errors.

Chapter 12: Product Interfaces

This section contains the following topics:

CA Workload Automation Restart Option for z/OS Schedulers

The interface supplied by CA WA performs the expiration of data sets residing on tape volumes under the control of CA TLMS. This action occurs when a tape data set is to be re-created by a job under the control of the CA WA rerun handler.

Installation

CA TLMS does not supply any modules in object or source for this interface. CA WA documentation should always be used as the final installation reference. The following is an overview of the installation process for planning purposes.

Reassembly of the user option table, U11OPT, is required with the following changes:

TLMS=YES

Specifies whether the CA TLMS - CA WA should be generated. The default is NO.

Include the ddname CAIVMFI to the U11RMS step JCL:

```
//CAIVMFI  DD DSN=CAI.TLMS.VMF,DISP=SHR
```

This interface allows CA-JCLCheck WA to access the CA TLMS Volume Master File (VMF) for retrieval of tape data set information. Part of CA-JCLCheck's reporting will include volume, volume sequence and DCB for each tape data set.

Installation

CA TLMS does not supply any modules in object or source for this interface. CA-JCLCheck WA documentation should be used as the installation reference for implementation of this interface.

CA ASM2 Interface

This interface allows CA ASM2 to tell CA TLMS when a volume is to be scratched by CA ASM2. Module TLMSASM2 has been designed to be called from CA ASM2, $NTEXIT and $FTEXIT.

Installation

This module uses the External Data Manager volume scratch program (TLMSSEDM) supplied as part of CA TLMS. This CA ASM2 user exit must be executed from a job or program identified as an EDM to CA TLMS.

Installation is accomplished using the following steps:

1. The load module should reside in the CA TLMS load library CAI.CTAPLINK.
2. Ensure the CA TLMS load library is concatenated to the jobs that will be calling both CA ASM2 exit points.
3. Activate the CA ASM2 user exits $NTEXIT and $FTEXIT according to the instructions that are supplied with CA ASM2.

More Information:

EDM System Option—External Data Manager (EDM) Pointer (see page 50)

CA Disk Interface

This interface allows CA Disk to tell CA TLMS when a volume owned by CA Disk is to be scratched. This module uses the External Data Manager (EDM) volume scratch program (TLMSSEDM) supplied as a part of CA TLMS.

Installation

Installation is accomplished using the following steps:

1. The load module should reside in the CA TLMS load library (CAI.CTAPLINK).
2. Ensure that the CA TLMS library is concatenated to the STEPLIBs for the CA Disk maintenance job STEPLIBs if CAI.CTAPLINK is not in the LNKLST.
3. Activate the CA Disk interface to CA TLMS through the CA Disk SYSPARMs. SYSPARM TMSCTLEX must be set to ADSTH017 and SYSPARM DYNEXPDT must be set to the default value of E99365. Refer to the TMSCTLEX exit documentation in the User Exit Descriptions section of the CA *Disk Systems Guide fo*r complete instructions on activating this interface using the ADSTH017 SYSPARM.

DFHSM Interface

This interface allows DFHSM to tell CA TLMS when a volume is to be scratched by DFHSM. CA TLMS distributes its version of module ARCTVEXT in CAI.CTAPLINK. ARCTVEXT is called as a user exit from DFHSM.

More Information:

EDM System Option—External Data Manager (EDM) Pointer (see page 50)
External Data Manager (EDM) Processing (see page 29)
External Data Manager Specifications (see page 159)

Installation

IBM no longer supplies the ARCTVEXT source and executable module. An object-code-only module is supplied by CA TLMS in the CTAPLINK library. You do not need to apply a USERMOD to the IBM SMP/E zones or libraries. To implement the TLMS supplied ARCTVEXT, perform the following steps:

1. Verify that the CA TLMS CTAPLINK library containing ARCTVEXT is either in the LINKLIST or in a STEPLIB data set in the DFHSM PROC.
2. Activate the scratch communication from DFHSM to CA TLMS using the following command:

   ```
   F DFHSM,SETSYS EXITON(TV)
   ```

3. Update the HSM parms member with the SETSYS command, so it is enabled at each start of DFHSM.

The source for the CA TLMS supplied ARCTVEXT is contained in the CTAPSAMP library in member ARCTVEXT. To modify it, use the JCL supplied in member TLMJU040 in CTAPJCL.

RACF Interface

Note: This interface is being retained from pre-Version 5.5 of CA TLMS for those data centers that elect not to use the Version 5.5 Security System interface, or wish to continue using the TAPEVOL class defined to RACF.

The CA TLMS RACF interface synchronizes the CA TLMS Volume Master File (VMF) with the RACF resource database. When a volume is scratched by CA TLMS, TLMSRACF automatically deletes the volume from the RACF data base, if it exists.

Installation

If user changes are needed to TLMSRACF, use TLMJU003 in CAI.CTAPJCL to apply USERMOD TLMU003. The resulting load module should reside in the CA TLMS load library CAI.CTAPLINK.

3480 Message Display Interface

This optional step provides for the installation of the 3480 message display exit. This applies only to a DFP system at Version 3.1 or higher. This exit is not supported in JES3 environments.

The 3480 message display exit is optional and, when installed into the operating system, provides the following two functions:

1. Displays the first eight characters of the CA TLMS z/OS scratch pool name on the 3480 display.
2. Disables an automatic cartridge loader, if present, when a scratch pool mount is requested.

Module IGXMSGEX is a CA TLMS user exit to the IBM module IGX00030, which is responsible for handling requests made through the MSGDISP macro. This exit must be linked with IGX00030 to be activated.

If the request is for a scratch pool defined in the CTOSCRxx member in CAI.CTAPOPTN, the automatic cartridge loader request will be disabled and the first eight bytes of the scratch pool name will be placed in the first eight bytes of the message area passed in the parameter list.

It is assumed that CTSMSGEX has already processed the mount request and placed the scratch pool number in the mount list element entry for the specified UCB. This number will be used to locate the correct scratch pool entry and extract the scratch pool name.

If it is desired that the automatic cartridge loader (ACL) *not* be disabled for scratch pool requests, then the source for this exit is provided (see source member CTSMSGEX in CAI.CTAPSAMP) and can be modified. To prevent disabling of the ACL, comment out the following instruction after the label SCRATCH:

```
NI  MSGFCB,255-MSGQACL  A. DISABLE AUTO CART LOADER
```

Installation

Use member CTSJUMSG from the CA TLMS CAI.CTAPJCL library as sample to apply USERMOD CTSUMSG.

3495 Basic Tape Library Data Server Support

Basic Tape Library Support (BTLS) for the IBM 3495 Tape Library Data Server is provided by a single module (CTSBTLS) which has been linked with the name of IDCLI04, the Set Category Installation Exit.

There are four basic functions available to support the 3495 BTLS:

1. Set cartridges entering the 3495 to the correct category.
2. Synchronize the CA TLMS VMF and the 3495 database.
3. Eject cartridges needed for off-site storage from the 3495.
4. Change the category of cartridges as they are scratched.

Two levels of support are available for the IBM 3495 Tape Library Data Server, depending on your operating system level.

Full IBM 3495 support is available if you are using at least the MVS/SP Version 4.3.0 and DFSMS/MVS 1.1.0. Basic Tape Library Data Server (BTLS) is for levels below this.

Installation

Use member CTSJUBTL from the CA TLMS CAI.CTAPJCL to apply USERMOD CTSUBTL.

3494, 3494/VTS, 3495 Tape Library Data Server Full Support

Full support for the IBM 3494, 3494/VTS, 3495 Tape Library Data Server is provided by a CA TLMS security exit modification, a CA TLMS supplied LCS exit module, and a utility synchronization program (CTSSYNC).

Two levels of support are available for the IBM 3495 Tape Library Data Server, depending on your operating system level.

Full IBM 3495 support is available if you are using at least the MVS/SP Version 4.3.0 and DFSMS/MVS 1.1.0. Basic Tape Library Support (BTLS) is for levels below this.

Installation

Use the following procedure to install the support:

1. Member CTSJUCBX in CAI.CTAPJCL contains sample JCL to apply USERMOD CTSUCBX. This USERMOD allows the customization and installation of OAM exit CBRUXENT, CBRUXEJC, and CBRUXVNL.
2. Verify that TLMSRACF has been modified to call the 3495 support for scratch notification. See CAI.CTAPJCL (TLMJU003).

Note: If there was a TLMSXUPD exit in the previous version, apply exit utilizing SMP/E and test the results.

CA OPS/MVS System State Manager

CA TLMS detects CA OPS/MVS automatically and communicates both active status events and heart beat events. The enabling technology for this is via a generic API call provided by CA OPS/MVS. Either CA TLMS or CA OPS/MVS can be started or stopped without harm.

Installation

No action is necessary. CA TLMS detects CA OPS/MVS automatically and opens an interface.

Failsafe USERMOD

This optional USERMOD can be installed into the operating system to prevent any tape processing from occurring when tape management intercepts are not applied. The Failsafe USERMOD will issue an error message and a WTOR each time a tape is mounted. This prevents each tape from being processed until an operator replies. The operator can allow the tape to process without tape management, cancel the job, or reject the tape.

Installation

Use member CTSJUSAF from TLMS CAI.CTAPJCL to apply USERMOD CTSUSAF.

An IPL is required to activate the Failsafe mode.

This will replace IBM's tape management exit IFG019VM in module IFG019RB which resides in SYS1.LPALIB. If you decide to put this USERMOD into MLPA instead, you must add the CAG8LIB2 DD statement to your CAS9 procedure.

Note: To remove the Failsafe code, remove the USERMOD with SMP/E RESTORE and IPL.

Disabling the JES3 Write Ring and Expiration Date Check

Because of the tape protection provided by CA TLMS, it is recommended that you disable two checks provided by JES3. During setup, JES3 checks the label expiration date of SL tapes mounted for scratch requests. All tapes originally created with a TLMS keyword expiration date will be rejected. JES3 allows this check to be disabled only on a job basis using the //*MAIN statement.

JES3 checks for the presence or absence of tape write rings. With CA TLMS, it is not necessary to remove the write rings.

The subsystem forces the label type and density of the tape being mounted to match the JCL by rejecting the tape if the label type and density do not match for JES3 users. For JES3 versions having IATUX62 (JES3 Version 2.1.5 or higher), this user exit allows SL functions to be overridden.

To disable both checks, modify the JES3 user exit IATUX29. See "Obtaining Job Accounting Information" in the *IBM JES3 SPL: User Modifications and JES3* Macros. Turn on the following bits in the JDAB control blocks (DSECTed by the JES3 macro IATYJDA):

```
01 JDABFLG3,JDABNOXP BYPASS EXPDT CHECK
01 JDABFLG2,JDABRNGC RING CHECK=NO
```

If the following code is not present in IATUX29, you will also need to add it prior to the t wo 01 instructions above.

```
USING IATISDT,R13
L     R8,JDABADDR
USING JDABSTRT,R8
```

You must also ensure that R15 is set with the correct value for the IATUX29 exit to be invoked. Please see the appropriate IBM manual.

Additionally, a suggested change is to the JES3 module IATIIDY to disable expiration date checking during dynamic allocation. Sequence numbers should be checked, as they may change. Consult your IBM PSR.

```
         TM  JSTDFLG1,JSTTA      Q. TAPE?              02580100
         BC  ALLOFF,SKIPNEXT     A. NO                 02580200
         OI  JSTHFLG4,JSTNOXP    BYPASS EXP CHK        02580300
SKIPNEXT OS  0H
```

CA Service Desk Integration

About CAISDI

A CA Common Services component, CAISDI (Service Desk Integration), enables CA mainframe z/OS products to automatically open request tickets in CA Service Desk for serious conditions that have been detected. You can optionally have the request tickets assigned to individuals of your choosing. Thus, important problems are brought to the attention of the selected individuals just seconds after they have been detected.

Along with CA TLMS, many other CA products are using the Service Desk Integration for centralizing the management of serious mainframe problems. A complete list of these products can be found in the *CA Common Services for z/OS Service Desk Integration Guide*.

The additional installation steps required for this feature are described in the *Installation Guide*. The CA TLMS's event definition members must be copied to the CAI.CTAPEVNT library and you must add a control statement to the CAISDI/els control parameter member, ELSSTART.

Each event member defines a specific event for CA TLMS. The event members contain the text to be used for the summary and description sections of the request ticket. Since these are ordinary source members, you can change the textual content to meet your specific requirements. In fact, the interface supports multiple languages, so you may translate the text into one of the supported languages if you wish. The details of customizing CAISDI for CA TLMS and other CA products are described in the *CA Common Services for z/OS Service Desk Integration Guide*.

The CAISDI API call is conditional, so it will do nothing until all of the elements are in place and CAIDSI is properly configured to communicate with your Service Desk server. Also, if there is a particular event for which you never want a request ticket opened, you may remove that event's definition member from the CAI.CTAPEVNT library with no adverse effect.

The CAISDI component also provides a batch utility that can artificially trigger one of CA TLMS's events. While this utility was designed for testing the CAISDI connections, you can also use it to trigger events of your own creation. Simply define new events for your own purposes and store the event member in CAI.CTAPEVNT. You can then use the utility as a conditional step in a JCL procedure to trigger the event you created.

CAISDI uses web services technology to access Service Desk. To activate the interface, you must have the following components installed and configured:

- CA Service Desk r11 server accessible to the mainframe using TCP/IP
- TCP/IP on your z/OS system
- CA Common Services r11 CAISDI on your z/OS system
- CA Common Services r11 CAICCI on your z/OS system

If you are interested in activating this interface, refer to the *CA Common Services for z/OS Service Desk Integration Guide*.

CAISDI Implementation for CA TLMS

When one of the following messages is issued, CA TLMS will call the CAISDI API to open a Service Desk (SD) request ticket:

```
CAT2401E CA TLMS -- AUXILIARY LIMIT REACHED *** EXPAND VMF IMMEDIATELY
CAT2402E CA TLMS - AUXILIARY THRESHOLD REACHED *** EXPAND VMF
CAT4501E INVALID CA TLMS RETENTION RUN DATE CARD
CAT4502E NO RECORDS ON CA TLMS VOLUME MASTER FILE
CAT4503E NO CONTROL RECORD ON CA TLMS VOLUME MASTER FILE
CAT4504E NO BASE RECORDS ON CA TLMS VOLUME MASTER FILE
CAT4505E NO RECORDS ON CA TLMS RETENTION MASTER FILE
CAT4506E NO LOCATION RECORDS CA TLMS RETENTION MASTER FILE
CAT4512E DEFAULT RECORD NOT IN RETENTION MASTER FILE
CAT4515E RETURN CODE FROM SORT NOT = TO ZEROS
CAT4516E ATTEMPTING TO FIND ALL VOLS. BASE VOL NOT FOUND
CAT4517E ATTEMPTING TO FIND ALL VOLS. ERROR READING CHAIN
CAT4521E LOCATION HAS NO MORE AVAILABLE STORAGE
CAT4522E INVALID REPORT CONTROL CARD
CAT4524E INVALID RETENTION TYPE PARAMETER IN CONTROL CARD
CAT4527E RETENTION RUN DATE > 3 DAYS ACCEPT KEYWORD REQUIRED
CAT4528E ENTRIES TO table EXCEED MAX OF nnnnn
CAT4533E INVALID OR REPETITIOUS INPUT CARD
CAT4534E INVALID LOCATION-id IN VMF (NOT DEFINED RMF LOC)
CAT4547E MULTI-VOL CHAIN ERROR BASE VOL=volser.
CAT4551E CAN NOT LOCATE THE TAPE STACKING POOL.
CAT9005E CA TLMS - VMF UPDATE FAILURE volser
CAT9010I CA TLMS INTERCEPTS NOT FULLY APPLIED. RESULTS MAY BE UNPREDICTABLE!!!
```

If the CAISDI component is installed and configured properly, a Service Desk request ticket will be opened for the problem that was detected. While the call to the CAISDI API will be made for each of these errors, the event definitions for many of these events include a MINTIME setting that may prevent additional request tickets from being created. For more information about MINTIME refer to the *CA Common Services for z/OS Service Desk Integration Guide*.

Unless you change the CA TLMS product control member, ATLCNTL, the request will be tied to a Service Desk software asset named "CA TLMS" and will be assigned to a pseudo contact named "System_TLMS_User." You can optionally change the ATLCNTL member to specify a real person as the assignee so the event will be directed immediately to someone on your staff.

Chapter 13: CA TLMS Health Checker

The IBM Health Checker for z/OS allows you to identify potential problems in your z/OS environment by checking system or product parameters and system status against recommended settings. The IBM Health Checker for z/OS is structured as a framework that includes a health check started task and various separate "check" routines provided by IBM or other vendors. CA TLMS provides health checks for CAI.CTAPOPTN settings, definitions or realtime processing conditions that could result in a problem. When these conditions are found the CA TLMS checks provide detailed recommendations on how to correct the problem. The CA TLMS health checks also make "best practice" recommendations for using CA TLMS.

To take advantage of this feature you must have the following components configured and running on your system:

- IBM Health Checker for z/OS - distributed with every supported level of z/OS
- CA Health Checker Common Service - a free CA Common Services component (no address space is required for the CA Health Checker Common Service)
- Common Tape System (CTS) address space - when started a health check subtask will always be started

CA TLMS provides the following health checks:

- TLMS_AUX_CUSHION_CRITICAL
- TLMS_AUX_CUSHION_WARNING
- TLMS_OPTION_NOTLMS
- TLMS_OPTION_PROTECT
- TLMS_OPTION_RECOVRY
- TLMS_OPTION_SECOPN
- TLMS_OPTION_SECURE
- TLMS_QUEUE_ACTIVE
- TLMS_VMF_ALOG_SEPERATION
- TLMS_VMF_UPDATE_NOT_POSSIBLE

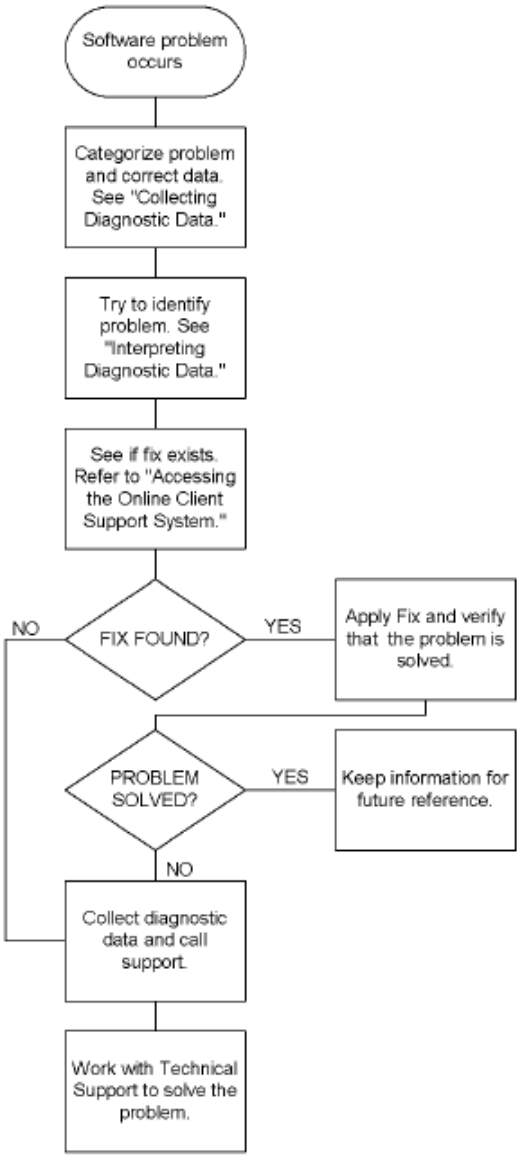
For more information about the use of the CA TLMS health checks, including check parameters that can be configured to change default thresholds or settings used in the checks, see Appendix D.

Chapter 14: Troubleshooting

This section contains the following topics:

Diagnostic Procedures

See the flowchart for a summary of the procedures to follow if you have a problem with a CA software product. Each of these procedures is described on the following pages.



Collecting Diagnostic Data

In the following table, the left column categorizes the problems. Follow the instructions in the right column to generate useful diagnostic data.

For problems in	Be sure to check
Installation	Your operating system type, maintenance level, and the options you established during installation. The SMP/E RECEIVE and APPLY output. Ensure that they completed successfully, with no unexplained linkage editor messages. That there are not mixed versions of CA TLMS. Check STEPLIB and LNKLST data sets.
Batch programs/utilities	The options in effect at the time of the problem (run CATOPTS). The SYSUDUMP/SYSABEND dump if the problem resulted in an abend. The CA TLMS reports, if any indicate the problem, and create a snapshot of volumes involved.
Online screens	Prints of screens involved, if possible. Otherwise, the sequence of keystrokes and fields modified. Snapshot of VMF records. Dump, if one was generated. (SYSUDUMP must be located.) CLIST used to invoke CATLISPF.
All areas	The SYSPRINT output from an execution of CATOPTS. Other vendor or IBM products that might affect the situation. All special PTFs, user exits/modifications and operating system intercepts that are installed and active. The combined console log for the period during which the problem occurred. Activity in other address spaces that may have a bearing on the problem (for example, tape mounts). Whether there was a system or user abend. If a user abend, please check user abend codes in the *Message Reference Guide.*

For problems in	Be sure to check
Online modules, online recorder, VMF updates	The options in effect at the time of the problem. Use TLMSSTAT. The JCL to execute CATOPTS is //OPTS EXEC CATOPTS. The SYSUDUMP/SYSABEND dump if the problem resulted in an abend, or the snap dump through the CTS command 'SPINOFF SNAP' or 'SPINOFF DUMP'. The execution JCL listing and cataloged procedures for the job involved, and the console log for the time involved. Allocation messages, job log, actual output of execution JCL. The CTS log for the time period can be obtained using the CTS command SPINOFF LOG. Type of tape involved: SL, BLP, NL, NSL, multivolume/multi-data set. Disposition of tape involved: NEW, MOD, PASS, OLD, and so on. Your operating system version number, and a snapshot of the volumes involved (see Chapter 6 of the *User Guide*), and DVs of the volumes before and after the job. The trace output for the period of the job and events causing the failure (see Tracing (see page 338)).

Tracing

The CA TLMS trace facility records every transaction before and after TLMS processing either manually or as a result of tape activity in another address space. Often, problems can be diagnosed and solved with the tracing output alone.

Interpreting Diagnostic Data

When you have collected the specified diagnostic data, write down your answers to the following questions:

1. What was the sequence of events prior to the error condition?
2. What circumstances existed when the problem occurred and what action did you take?
3. Has this situation occurred before? What was different then?

4. Did the problem occur after a particular PTF was applied or after a new version of the software was installed?
5. Have you recently installed a new version of the operating system?
6. Has the hardware configuration (tape drives, disk drives, and so forth) changed?

From your response to these questions and the diagnostic data, try to identify the cause and resolve the problem.

Accessing the Online Client Support System

CA is making extensive use of the Internet for your benefit. CA encourages you to "surf the net" to the CA home page at **www.ca.com** and the online support at **http://ca.com/support**. The CA Internet site provides a great variety of information about CA products and services, including:

- Service and support
- Product information and sales
- CA-World conference information
- Press releases
- CA user groups

SupportConnect gives you real time, interactive access to CA product support information through the Internet. Using SupportConnect, you can:

- Open new issues
- Browse or update your existing issues and enhancement requests
- Perform keyword searches
- Download solutions, PTFs, and important notices regarding CA products, maintenance, and documentation

Requirements for Using SupportConnect

The following are the requirements to use SupportConnect:

- You must be a CA client with a current maintenance agreement.
- You must register through the CA Internet site.
- You must access the Internet with a browser that supports the HTML specification 2.0 or higher, such as Netscape Navigator 4.0 or higher or Microsoft Internet Explorer 3.0 or higher.

  Browsers that meet the HTML requirement support the following functions, which are required for SupportConnect:

  - Secure sockets layer (SSL) to encrypt your transaction traffic
  - Encrypted data records (known as COOKIES)
  - HTML tables

SupportConnect Security

SupportConnect runs as a secured server (SSL). You may need to configure your browser to enable SSL. Guidelines for doing this are provided on the CA Technical Support page.

SupportConnect Functions

With SupportConnect, and also its companions CustomerConnect and AccountConnect, you can perform the following:

- Open a new issue

  Open an issue for, or request an enhancement to, one of your CA products.

- Browse your issues and enhancement requests

  Display all issues for your site. The issues are grouped into three categories: Open, Closed, and Enhancement Requests (DARs).

- Browse and/or download solutions

  Specify criteria for selecting solutions, which you can then view or download.

- Search the CA knowledge base

  Specify criteria for searching the CA database for solutions, problems, and keywords that can provide you with immediate answers to your product support questions and concerns.

- Update your SupportConnect profile

  Make changes to your default email address, phone number, and password whenever necessary.

- Display your site's licenses

  View a list of all the CA products for which your company site is currently licensed.

- Display SupportConnect news items

  View and download recently published solutions for CA products, instructions for downloading from SupportConnect, and helpful information for using CA-StarTrak, SupportConnect, or other CA products.

Accessing the Technical Support Phone Services Directory

The CA Technical Support Phone Services Directory lists each CA product and the telephone number to call for primary support for that product. To access the Support Phone Services Directory, set your browser for SupportConnect.ca.com and click on Contact Us.

Calling CA Technical Support

For online technical assistance and a complete list of locations and phone numbers, contact Technical Support at http://ca.com/support. Technical Support is available 24 hours a day, 7 days a week. For telephone assistance, call:

U.S. and Canada

1-800-225-5224

International

(1) 631-881-6801

If you are unable to resolve the problem, please have the following information ready before contacting CA Technical Support:

- All the diagnostic information described in Collecting Diagnostic Data
- Product name, version number, operating system, and genlevel.
- Product name and version number of any other software you suspect is involved.
- Version level and PUTLEVEL of the operating system.
- Your name, telephone number and extension (if any).
- Your company name.

- Your site ID.
- A severity code. This is a number (from 1 to 4) that you assign to the problem. Use the following to determine the severity of the problem:

  1

  a "system down" or inoperative condition

  2

  a suspected high-impact condition associated with the product

  3

  a question concerning product performance or an intermittent low-impact condition associated with the product

  4

  a question concerning general product utilization or implementation

Product Versions and Maintenance

Clients are requested to operate only under currently supported versions of the product.

Clients with current maintenance agreements also receive ongoing product maintenance. When a new version of the system is available, a notice is sent to all current clients.

Requesting Enhancements

CA welcomes your suggestions for product enhancements. All suggestions are considered and acknowledged. You can use either of two methods to request enhancements:

- Contact your Account Manager who will initiate a Demand Analysis Request (DAR) for you.
- Enter your request through SupportConnect.ca.com,, the CA web-based, interactive support system.

Generating a Problem Report

Once a CA Technical Support representative has determined that your problem requires further investigation, you can use the following to generate a problem report.

TLMSSTAT Utility

The TLMSSTAT diagnostic facility produces a problem report for you to fill out and send in with all problem documentation.

To invoke TLMSSTAT, execute the CATOPTS proc in your sample JCL library:

```
//TMSSTATS EXEC CATOPTS
```

Edit the JCL to your installation's standards, and submit the job.

The last page of the problem report provides a status of the CA products that you have installed. In most instances, this report includes the product version number, gen level, and installation options. This is a sample of a report for CA TLMS:

```
THURSDAY, MARCH 25, 2010.084                    TLMSSTAT    JOB=CRAJOSTS  STEP=TMSSTATS    TIME=10:05:03                PAGE=00001
 CA TLMS OPTIONS LISTING
    CA TLMS           NN.n           GENLEVEL 1003TLC00
 GENERAL:
    ALTCTR=D1        COMPANY=YOUR COMPANY NAME
    DATACTR=DC       DATEFMT=MM/DD/YYYY               PAGESIZ=58
    VMFNAM =ASM.TLMSDV.VMF
    VMFINAM=ASM.TLMSDV.VMFINDEX
 TECHNICAL:
    FORSPEC=NO       LOGID=240        LBLDTRY=YES
    QSIZE=064        MSGPFX=CTS       SMS=NO
    NSM=BC           SCR=BC           EDM=BC
    ROUTAUX=NO
    ROUTINQ=YES    ( RT=14      UPD=YES    MSG=DEL                )
 PROCESSING:
    ABEND=24         BRKCHN=OPEN      CATDAYS=1        DISP=OUTPUT
    DBLDRIV=NO       DBLTIME=000000   IDSNVER=YES
    INPUT=YES      ( CHN=NO     BLP=YES    NL=YES     SL=YES       )
    KDATE=MAX        MANUAL=NO        NOTLMS=ABEND     PROTECT=ALL
    RECOVRY=ALTLOG   SCRATCH=CDS      SERIND=YES
    UNCATLG=YES      VSNPAD=          VSNREQD=YES
 TLMSHTAB:
    MSG  ID.         OPTION
    CAT9010P         SUP
    CAT9013P         SUP
```

```
THURSDAY, MARCH 25, 2010.084                    TLMSSTAT    JOB=CRAJOSTS  STEP=TMSSTATS    TIME=10:05:03                PAGE=00002
                          CTS MESSAGE INTERCEPT TABLE ENTRIES CURRENTLY ACTIVE

            MESSAGE          PRODUCT IDENTIFICATION
            IAT5210          CA TLMS       TAPE MANAGEMENT SYSTEM
            IAT5410          CA TLMS       TAPE MANAGEMENT SYSTEM
            IAT5624          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC101A          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC400A          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC404E          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC501A          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC501E          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC502E          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC507D          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC534D          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEC704A          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEF233A          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEF233D          CA TLMS       TAPE MANAGEMENT SYSTEM
            IEF234E          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB101A          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB501A          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB501E          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB502E          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB507D          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB534D          CA TLMS       TAPE MANAGEMENT SYSTEM
            JBB704A          CA TLMS       TAPE MANAGEMENT SYSTEM
            JDJ233A          CA TLMS       TAPE MANAGEMENT SYSTEM
            JDJ233D          CA TLMS       TAPE MANAGEMENT SYSTEM
            JDJ234E          CA TLMS       TAPE MANAGEMENT SYSTEM
```

```
THURSDAY, MARCH 25, 2010.084                    TLMSSTAT    JOB=CRAJOSTS  STEP=TMSSTATS    TIME=10:05:03                PAGE=00003
                         SCRATCH POOL DEFINITION REPORT FOR CA TLMS       TAPE MANAGEMENT SYSTEM

         SCRATCH POOL(S) DEFINED IN PPOPTION MEMBER: CTOSCRBC


          SCRATCH POOL ID          VOLUME SERIAL RANGE(S)

           VTAPE                    001500-001540
           T3590                    TP0000-TP0099
           XCARTS                   CT0000-CT0099
           HIVOLS                   001541-001599

  END OF SCRATCH POOL DEFINITION REPORT
```

```
THURSDAY, MARCH 25, 2010.084                    TLMSSTAT    JOB=CRAJOSTS  STEP=TMSSTATS    TIME=10:05:03                PAGE=00004
                         SCRATCH POOL ASSIGNMENT REPORT FOR CA TLMS       TAPE MANAGEMENT SYSTEM

         NON-SPECIFIC MOUNT RULES ASSIGNED IN PPOPTION MEMBER: CTONSMBC


 MANAGEMENT CLASS / EDM NAME / DATASET NAME                        JOBNAME      LOW DATE   HIGH DATE  UNIT SCRATCH POOL

 OT23A.STK3590.-                                                   -            N/A        N/A        -    T3590
 SYS2.ARC.-                                                        -            N/A        N/A        -    T3590
 SYS3.BKUP.-                                                       -            N/A        N/A        -    HIVOLS
 -                                                                 DISTR-       N/A        N/A        -    XCARTS
 -                                                                 -            N/A        N/A        -    VTAPE

  END OF SCRATCH POOL ASSIGNMENT REPORT
```

```
THURSDAY, MARCH 25, 2010.084                    TLMSSTAT    JOB=CRAJOSTS  STEP=TMSSTATS    TIME=10:05:03                PAGE=00005
                        EDM RULE ASSIGNMENT REPORT FOR CA TLMS       TAPE MANAGEMENT SYSTEM

         EXTERNAL DATA MANAGER RULES ASSIGNED IN PPOPTION MEMBER: CTOEDMBC


  EDMID    DATA SET NAME                                PROGRAM NAME   JOBNAME NAME   DD NAME

  TECHS    TECH.LAB020.DATA                             -              -              -
  TECHS    TECH.LAB010.DATA                             -              -              -
  DISK     -                                            ADSMI-         -              -
  DISK     -                                            ADSMI002       -              -
  HSM      -                                            ARCCTL         -              -

 END OF EDM RULE ASSIGNMENT REPORT
```

Appendix A: TLMDATE Macro - Common Date Processing Routines

The TLMDATE macro is a general use macro which provides a common set of routines for manipulating CA TLMS keywords, and internal and external dates in various formats. TLMDATE accepts dates specified in the range January 0, 1960 (1960/000) to December 32, 2155 (2155/366).

This section contains the following topics:

Date Conversion Routines

The TLMDATE macro furnishes several routines to facilitate the conversion of specific date formats to and from the CA TLMS internal date format. Those formats supported are:

JFCB

The format for this internal date is X'yydddd', where yy is the year in hexadecimal (X'63'=1999, X'64'=2000 and so on, and dddd is the day of the year in hexadecimal.

HDR1

This is the date as stored in the FL1EXPDT field of the HDR1 label. Its format is C'cyyddd', where c is the century (blank=1900, 0=2000, 1=2100 and so on).

CA TLMS External Format

This refers to one of the date formats supported by CA TLMS for display and input purposes.

Those TLMDATE macros that are used for date conversion operations require that the following parameters be defined:

FRDATE=

Points to the field to be converted.

TODATE=

Points to the field that receives the resulting converted date.

Date Calculation Routines

Several forms of the TLMDATE macro are provided to perform ADD TO DATE and SUBTRACT FROM DATE operations, and to calculate the days difference between two dates. This series of macros only works with CA TLMS internal dates. For these date calculation routines, the following parameters must be defined:

TODATE=

Defines the CA TLMS internal date to be added to, or is the HIGH DATE when calculating the difference between two dates.

FRDATE=

Defines the CA TLMS internal date to be subtracted from, or is the LOW DATE when calculating the difference between two dates.

LOAD

Specifies whether the DATE module is to be loaded or which LPA version is to be used.

DAYS=

Indicates the number of days to be added to or subtracted from a date, or points to the field to receive the result when calculating the difference between two dates.

YEARS=

Indicates the number of years to be added to or subtracted from a date.

RESULT=

Points to the field to receive the date resulting from an add or subtract operation. RESULT is always a CA TLMS date, internal packed 4 bytes.

CA TLMS Internal Date Format

All internal date fields in CA TLMS are stored as 4 bytes, packed length. The format of the internal date is ccyyddds, where ccyy is the century and year combined (0099=1999, 0100=2000 and so on, ddd is the Julian day of the year, and s is the sign (always positive). CA TLMS does not attempt to maintain the sign as a particular value, but guarantees the value to be positive.

Internal dates can be manipulated by the TLMDATE macro. The chart provided in Appendix C of this guide lists each CA TLMS expiration date keyword, and the equivalent internal date stored by CA TLMS.

TLMDATE Macro Functions

The various forms of the TLMDATE macro are detailed on the following pages in alphabetical order by function type. They are:

ADD_DAYS

Add a number of days to a date

ADD_WORK

Add a number of work days to a date (adjusted for weekends)

ADD_YEARS

Add a number of years to a date

ANCHOR

Define the Anchor Control Block

CLOSE

Close date routine processing

EXPLODED_DATA

Generate storage area for date related information after a call to TLMDATE TO_EXPLODED

FROM_EXTERNAL

Convert an external date to a CA TLMS internal date

FROM_HDR1

Convert HDR1 date to a CA TLMS internal date

FROM_JFCB

Convert JFCB date to a CA TLMS internal date

NUM_DAYS

Calculate and store days difference between two dates

OPEN

Initialize date processing routines

PARM

Define parameter list for CTSDATE module

RETURN_DATE

Return current date and time

RETURN_TYPE

Determine type of CA TLMS internal date (calendar or CA TLMS keyword)

SET_FORMAT

Set the preferred date format

SET_KEYWORD

Define a keyword value

SUB_DAYS

Subtract a number of days from a date

SUB_YEARS

Subtract a number of years from a date

TEST_FORMAT

Determine date format validity

TO_EXPLODED

Convert a CA TLMS internal date to exploded format

TO_EXTERNAL

Convert from a CA TLMS internal date to an external date with format

TO_HDR1

Convert a CA TLMS internal date to HDR1 date format

TO_JFCB

Convert a CA TLMS internal date to JFCB date format

TO_PREFERRED

Convert a CA TLMS internal date to preferred date format

Note: The OPEN form of the TLMDATE macro initializes the date processing routines, and must be issued before any other forms of the macro can be successfully executed.

Coding the TLMDATE Macros

A thorough knowledge of IBM Assembler Language and macro processing standards is assumed.

Use the following coding conventions when coding the TLMDATE macros:

- The label name must start in column 1.
- The macro name must be followed by at least one space.
- The character X in column 72 indicates that a continuation follows.
- Continuation statements must start in column 16.

Where macro examples are supplied in this chapter, the macro name is assumed to begin in column 10.

The following pages present the TLMDATE macros in alphabetical order by function type.

TLMDATE ADD_DAYS Macro—Add Days to a Date

Use the ADD_DAYS form of the TLMDATE macro to add a number of days to a date. This function adds DAYS to TODATE and places the resulting date into RESULT.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE ADD_DAYS
    ,TODATE=internal1
    ,DAYS=days
    ,RESULT=internal2
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

TODATE

Specifies the CA TLMS internal date to be added to. This field cannot contain a CA TLMS keyword.

Type: RX-type label, (Rn), internal packed (4 bytes)

DAYS

Specifies the number of days to be added to TODATE.

Type: RX-type label, (Rn), packed (4 bytes), or a constant (quotes)

RESULT

Specifies the field to receive the resulting CA TLMS internal date.

Type: RX-type label, internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE ADD_DAYS Macro

```
                                                          Pos. 72
                                                          |
TLMDATE  ADD_DAYS,TODATE=DATE1,RESULT=DATE3,DAYS=PACK1,   X
      ANCHOR=ANCHOR,PARM=DATEPARM
```

TLMDATE ADD_WORK Macro—Add Work Days to a Date

Use the ADD_WORK form of the TLMDATE macro to add a number of work days to a date. This function adds DAYS to TODATE and places the resulting date into RESULT. The DAYS value is adjusted to accommodate weekends.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE ADD_WORK
    ,TODATE=internal1
    ,DAYS=days
    ,RESULT=internal2
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

TODATE

Specifies the CA TLMS internal date to be added to. This field cannot contain a CA TLMS keyword.

Type: RX-type label, (Rn), internal packed (4 bytes)

DAYS

Specifies the number of days to be added to TODATE.

Type: RX-type label, (Rn), packed (4 bytes), or a constant (quotes)

RESULT

Specifies the field to receive the resulting CA TLMS internal date.

Type: RX-type label, internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE ADD_WORK Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  ADD_WORK,ANCHOR=ANCHOR,PARM=DATEPARM,          X
      TODATE=TCRTDT,DAYS=PRODATE,RESULT=TEXPDT
```

TLMDATE ADD_YEARS Macro—Add Years to a Date

Use the ADD_YEARS form of the TLMDATE macro to add a number of years to a date. This function adds YEARS to TODATE and places the resulting date into RESULT.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE ADD_YEARS
    ,TODATE=internal1
    ,YEARS=years
    ,RESULT=internal2
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

TODATE

Specifies the CA TLMS internal date to be added to. This field cannot contain a CA TLMS keyword.

Type: RX-type label, (Rn), internal packed (4 bytes)

YEARS=*years*

Specifies the number of years to be added to TODATE.

Type: RX-type label, (Rn), packed (4 bytes)

RESULT

Specifies the field to receive the resulting CA TLMS internal date.

Type: RX-type label, internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE ADD_YEARS Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  ADD_YEARS,YEARS=NUMYEARS,TODATE=DATE,          X
      RESULT=DATEPYR,ANCHOR=R1ANCR,PARM=R1PARMS,        X
      NORMAL=ADDOKAY
```

TLMDATE ANCHOR Macro—Define Anchor Control Block

Use the ANCHOR form of the TLMDATE macro to define the Anchor Control Block.

At least one TLMDATE ANCHOR function must be coded to allow PREFIX to default. Other forms of the TLMDATE macro rely on the labels generated to obtain offsets.

This macro has the following format:

label TLMDATE ANCHOR [,PREFIX=*xx*]

PREFIX=*xx*

(Optional) Specifies the prefix used for the work area.

Default: DA (Date Anchor)

The following is the data area generated for TLMDATE ANCHOR:

```
         DS    0F
name     DS    0XL32             ANCHOR CONTROL BLOCK
DATOKEN  DC    A(0)              TOKEN (POINTER TO WORKAREA)
DAFUNC   DC    A(0)              FUNCTION TO BE PERFORMED
DAFNOPEN EQU   4                 OPEN
DAFNSEFM EQU   8                 SET_FORMAT
DAFNTEFM EQU   12                TEST_FORMAT
DAFNCLOS EQU   16                CLOSE
DAFNRTDT EQU   20                RETURN_DATE
DAFNRTTY EQU   24                RETURN_TYPE
DAFNNMDA EQU   28                NUM_DAYS
DAFNSBDA EQU   32                SUB_DAYS
DAFNADDA EQU   36                ADD_DAYS
DAFNSBYR EQU   40                SUB_YEARS
DAFNADYR EQU   44                ADD_YEARS
DAFNADWK EQU   48                ADD_WORK
DAFNFRJF EQU   52                FROM_JFCB
DAFNTOJF EQU   56                TO_JFCB
DAFNFRHD EQU   60                FROM_HDR1
DAFNTOHD EQU   64                TO_HDR1
DAFNFRXT EQU   68                FROM_EXTERNAL
DAFNTOXT EQU   72                TO_EXTERNAL
DAFNTOPR EQU   76                TO_PREFERRED
DAFNTOXP EQU   80                TO_EXPLODED
DAFNKYCN EQU   84                KEYWORD_CONFLICTS
DAFNSEKY EQU   88                SET_KEYWORD
DARETURN DC    A(0)              RETURN CODE
DARCNORM EQU   0
DARCWARN EQU   4
DARCERRO EQU   8
DARCFAIL EQU   12
```

```
DAREASON DC    A(0)              REASON CODE (RC=4,8,12 ONLY)
DARSDFLT EQU   0                 DEFAULT. NO ADDITIONAL INFO
DARSNTPK EQU   4           RC=8  DATE FIELD NOT PACKED
DARSBDKY EQU   4           RC=12 KEY= NOT 'YES' OR 'NO '
DARSBDSQ EQU   8           RC=12 WRONG SEQUENCE FOR FUNCTIONS
DARSDTRG EQU   8           RC=8  NOT IN ACCEPTABLE DATE RANGE
DARSUKWD EQU   12          RC=8  UNDEFINED KEYWORD
DARSBNRG EQU   16          RC=8  BINARY VALUE NOT IN VALID RANGE
DARSDTCM EQU   20          RC=8  COMPUTED DATE NOT IN RANGE
DARSKYDD EQU   24          RC=8  DDD FOR KEYWORD NOT IN RANGE
DARSDTDD EQU   28          RC=8  DDD FOR DATE NOT IN RANGE
DARSBDFM EQU   32          RC=8  INVALID FORMAT PARM
DARSUNFM EQU   36          RC=8  EXT DATE IN UNKNOWN FORMAT(WONT MAP)
DARSKWDU EQU   40          RC=8  KWD USE NOT APPROPRIATE FOR FUNC
DARSBDHD EQU   44          RC=8  NOT VALID HDR1 DATE
DARSBDDD EQU   48          RC=8  DD INVALID FOR MM
DARSBDMM EQU   52          RC=8  MM INVALID (NOT 1 -12)
DARSBDCM EQU   56          RC=8  MMM INVALID (NOT VALID ABBREV)
DAFDBK   DC    A(0)               FEED BACK (RC=0 ONLY)
DAFBZERO EQU   0                    ZEROS
DAFBJDAT EQU   4                    JDATE       - INTERNAL DATE
DAFBOJDT EQU   8                    OJDATE      - OLD JULIAN FMT
DAFBFRGN EQU   12                   FOREIGN     - NON CA TLMS
DAFBLDAT EQU   16                   LDATE/DDD   - LAST USE  RETENTION
DAFBCTLG EQU   20                   CATLG       - CATALOG   RETENTION
DAFBCYCL EQU   24                   CYCLE/CCC   - CYCLE     RETENTION
DAFBPERM EQU   28                   PERM        - PERMANENT RETENTION
DAFBCTDD EQU   32                   CATLG/DDD   - CATLG/DDD RETENTION
DAFBUSER EQU   36                   USER/NNN    - USER RETENTION
DAFBSTAT EQU   40                   STATS/NNN   - HOLD STATUS CODE
DAFBMSG  EQU   44                   MSG/NNN     - AUXILLARY MESSAGE
DAFBAGE  EQU   48                   AGE/DDD
DALEVEL  DC    CL8'BASE'          MACRO LEVEL
DAMODULE DC    A(0)               ADDRESS OF DATE MODULE
```

TLMDATE CLOSE Macro—Close Date Processing

Use the CLOSE form of the TLMDATE macro to close date routine processing.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE CLOSE
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,FAIL=branch]
```

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE CLOSE Macro

```
TLMDATE  CLOSE,ANCHOR=ANCHOR,PARM=DLIST
```

TLMDATE EXPLODED_DATA Macro—Generate Data Area

Use the EXPLODED_DATA form of the TLMDATE macro to generate the data area in which all date related information is stored after a call to TLMDATE TO_EXPLODED.

This macro has the following format:

label TLMDATE EXPLODED_DATA [,PREFIX=*xx*]

PREFIX=*xx*

(Optional) Specifies the prefix used for the work area.

Default: DE (Date Exploded)

The following is the data area generated for TLMDATE EXPLODED_DATA:

```
         DS    0F
name     DS    0CL64              EXPLODED DATE CONTROL BLOCK
DETYPDAT DC    F'0'               TYPE OF DATE IN BINARY
DETYPNAM DC    CL10' '            NAME ASSOCIATED WITH DATE TYPE
DETYPALT DC    CL06' '            ALTERNATE SHORT KEYWORD NAME
DEDAYNAM DC    CL10' '            CHARACTER DAY-OF-WEEK
DEDAYABV DC    CL03' '            CHARACTER DAY-OF-WEEK (ABBREV.)
DEDOW    DC    CL01'0'            NUMERIC DAY-OF-WEEK (1=SUN-7=SAT)
DEDAYTYP DC    CL01' '            DAY TYPE (H)OLIDAY,(W)ORK,WEEK(E)ND
DELYR    DC    CL01'0'            LEAP YEAR INDICATOR (1=LEAP)
DEPAKDDD DC    PL04'0'            PACKED JULIAN DAY   (000-366)
DEMONNAM DC    CL10' '            CHARACTER MONTH NAME
DEMONABV DC    CL03' '            CHARACTER MONTH NAME (ABBREV.)
DEALLDAT DC    0CL11' '           ALL DATE (GREGORIAN AND JULIAN)
DEMM     DC    CL02' '            NUMERIC MONTH       (01-12)
DEDD     DC    CL02' '            NUMERIC DAY         (00-32)
DEYYYY   DC    CL04' '            NUMERIC YEAR        (1960-2155)
DEDDD    DC    CL03' '            NUMERIC JULIAN DAY  (000-366)
```

TLMDATE FROM_EXTERNAL Macro—Convert to Internal Date

Use the FROM_EXTERNAL form of the TLMDATE macro to convert a CA TLMS external date to an internal date.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE FROM_EXTERNAL
    ,FRDATE=external
    ,TODATE=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,ZEROS=branch]
    [,JDATE=branch]
    [,OJDATE=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,STATS=branch]
    [,AGE=branch]
    [,MSG=branch]
```

FRDATE=*external*

Specifies the external date to be converted.

Type: RX-type label, (Rn), external (10 bytes).

TODATE=*internal*

Specifies the date to receive the translated internal date.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If none are defined, the default is to NORMAL.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

OJDATE=*branch*

(Optional) Specifies the label to get control if the resulting date was converted from the YYDDD format. If not specified, JDATE is used.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label

Example: TLMDATE FROM_EXTERNAL Macro

```
                                                        Pos. 72
                                                        |
TLMDATE   FROM_EXTERNAL,FRDATE=EXTERNAL,TODATE=DLOW,    X
      ANCHOR=ANCHOR,PARM=DLIST,ERROR=ERROR02
```

TLMDATE FROM_HDR1 Macro—Convert Date to Internal Format

Use the FROM_HDR1 form of the TLMDATE macro to convert a date in HDR1 format to a date in CA TLMS internal format. This function translates FRDATE (HDR1 format) into TODATE (CA TLMS internal format).

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE FROM_HDR1
    ,FRDATE=hdrldate
    ,TODATE=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

[,ZEROS=*branch*]

[,JDATE=*branch*]

[,PERM=*branch*]

FRDATE=*hdrldate*

Specifies the date to be converted.

Type: RX-type label, HDR1 format (6 bytes)

TODATE=*internal*

Specifies the date to receive the translated internal date.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If neither is specified, the default is to NORMAL.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branch is performed only if the return from CTSDATE is NORMAL. It represents dates that are CA TLMS keywords.

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

Example: TLMDATE FROM_HDR1 Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  FROM_HDR1,FRDATE=FL1EXPDT,TODATE=XEXPDT,       X
      ANCHOR=XWANCHOR,PARM=XWPARM
```

TLMDATE FROM_JFCB Macro—Convert Date to Internal Format

Use the FROM_JFCB form of the TLMDATE macro to convert a date in JFCB format to a date in CA TLMS internal format. This function translates FRDATE (JFCB format) into TODATE (CA TLMS internal format).

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE FROM_JCFB
    ,FRDATE=jfcbdate
    ,TODATE=internal
    ,KEY=[YES|NO|LABEL]
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,ZEROS=branch]
    [,JDATE=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,MSG=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,AGE=branch]
```

FRDATE=*jfcbdate*

Specifies the date to be converted.

Type: RX-type label, JFCB format, hexadecimal (3 bytes)

TODATE=*internal*

Specifies the date to receive the translated internal date.

Type: RX-type label, (Rn), internal packed (4 bytes)

KEY=[YES|NO|LABEL]

Indicates whether the date being translated can contain keywords (97000-99366).

Valid values:YES, NO, or a field containing the value

Type: RX-type label, (Rn) or constant (quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If they are omitted, NORMAL is assumed. If neither is specified, the default is to NORMAL.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label

Example: TLMDATE FROM_JFCB Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  FROM_JFCB,FRDATE=JFCBCRDT,TODATE=TMCRTDT,      X
      KEY='NO',ANCHOR=XWANCHOR,PARM=XWPARM,             X
      ERROR=ABEND10,FAIL=ABEND10
```

TLMDATE NUM_DAYS Macro—Days Between Dates

Use the NUM_DAYS form of the TLMDATE macro to calculate and store the number of days between two dates. This function subtracts TODATE from FRDATE and places the difference (number of days) into DAYS. The DAYS= result excludes one day from the FRDATE-TODATE range.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE NUM_DAYS
    ,FRDATE=internal1
    ,TODATE=internal2
    ,DAYS=days
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

FRDATE=*internal1*

Specifies the internal low date used in the subtraction.

Type: RX-type label, (Rn), internal packed (4 bytes)

TODATE=*internal2*

Specifies the internal high date used in the subtraction.

Type: RX-type label, (Rn), internal packed (4 bytes)

DAYS=*days*

Specifies the field to receive the resulting difference.

Type: RX-type label, (Rn), packed (4 bytes) or a constant (quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE NUM_DAYS Macro

```
                                                    Pos. 72
                                                    |
TLMDATE  NUM_DAYS,FRDATE=XCRTDT,TODATE=XEXPDT,      X
      DAYS=XDAYS,ANCHOR=XWANCHOR,PARM=XWPARM,       X
      ERROR=ABEND10,FAIL=ABEND10
```

TLMDATE OPEN Macro—Initialize Date Routines

Use the OPEN form of TLMDATE to initialize date routines for processing. This includes GETMAINing storage and obtaining the Preferred date format. TLMDATE OPEN must be issued before any other forms of the macro will execute successfully. OPEN establishes the ANCHOR used by all other calls.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information into the macro.

This macro has the following format:

```
label TLMDATE OPEN ,FMT=format
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

FMT=*format*

Specifies a 10-byte field containing a date pattern.

Type: RX-type label, (Rn), or constant (quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE OPEN Macro

```
                                                                Pos. 72
                                                                |
TLMDATE  OPEN,FMT=DATEFMT,ANCHOR=DAANCHOR,PARM=DATEPARM,        X
      ERROR=ABEND10,FAIL=ABEND10
```

TLMDATE PARM Macro—Define Parameter List

Use the PARM form of the TLMDATE macro to define a parameter list sufficient to hold the parameters passed to CTSDATE.

This macro has the following format:

label TLMDATE PARM

TLMDATE RETURN_DATE Macro—Return Date and Time

Use the RETURN_DATE form of the TLMDATE macro to return the current date and time.

This macro has the following format:

```
label TLMDATE RETURN_DATE
    ,DATE=internal
    ,TIME=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

DATE=*internal*

Specifies the field to receive the current date.

Type: RX-type label or (Rn), internal packed (4 bytes)

TIME=*internal*

Specifies the field to receive the current time.

Type: RX-type label or (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE RETURN_DATE Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  RETURN_DATE,DATE=CURRDATE,TIME=CURRTIME,       X
      ANCHOR=ANCHOR,PARM=DPARM
```

TLMDATE RETURN_TYPE Macro—Return Internal Date Type

Use the RETURN_TYPE form of the TLMDATE macro to return the type of the internal date.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE RETURN_TYPE
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,STATS=branch]
    [,AGE=branch]
    [,MSG=branch]
```

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If neither is specified, the default is to NORMAL.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label.

Example: TLMDATE RETURN_TYPE Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  RETURN_TYPE,ANCHOR=ANCHOR,PARM=DATEPARM,       X
      JDATE=NON98,KEYWORD=NON98,LDATE=EXTN98,           X
      DATE=TMEXPDT
```

TLMDATE SET_FORMAT Macro—Set Date Format

Use the SET_FORMAT form of the TLMDATE macro to establish the preferred date format.

This macro has the following format:

```
label TLMDATE SET_FORMAT
    ,FMT=format
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

FMT=*format*

Specifies a 10-byte field containing a date pattern.

Type: RX-type label, (Rn), or constant (quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE SET_FORMAT Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  SET_FORMAT,FMT=DATEFMT,ANCHOR=R1ANCR,          X
      PARM=R1PARMS,NORMAL=PARMDAT5
```

TLMDATE SET_KEYWORD Macro—Define Keyword Value

Use the SET_KEYWORD form of the TLMDATE macro to define a keyword value.

This macro has the following format:

```
label TLMDATE SET_KEYWORD
    ,TODATE=internal1
    ,KEYWORD=keyword
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,VALUE=value]
```

TODATE=*internal1*

Specifies an internal date to be set to the keyword specified by the KEYWORD= operand.

KEYWORD=*keyword*

Represents the CA TLMS keyword to which TODATE will be set (CATALOG, LDATE and so on). When specified as a constant, the value coded is the actual keyword (KEYWORD='LDATE'). When specified as an RX-type label or (Rn), KEYWORD points to a fullword containing the feedback code associated with the keyword, such as A(DAFBLDAT) for LDATE. (See TLMDATE_ANCHOR.)

Type: RX-type label, (Rn) or constant (in single quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

VALUE

Specifies a number to be associated with the keywords ZEROS, LDATE, CYCLE, CATLG, USER, FOREIGN, PERM and STATS. VALUE= cannot be coded if the KEYWORD= value is coded as a constant and the keyword specified is not a value. For example, KEYWORD='PERM' and VALUE= are invalid. VALUE= is not validated if KEYWORD= is specified as an RX-type label or (Rn) value.

Type: RX-type label, (Rn) or constant (in single quotes), packed (4 bytes)

Example: TLMDATE SET_KEYWORD Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  SET_KEYWORD,KEYWORD='PERM',TODATE=TMEXPDT,     X
      ANCHOR=XWANCHOR,PARM=XWPARM
```

TLMDATE SUB_DAYS Macro—Subtract Days From Date

Use the SUB_DAYS form of the TLMDATE macro to subtract a number of days from a date. This function subtracts DAYS from FRDATE and places the resulting date into RESULT.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE SUB_DAYS
    ,FRDATE=internal1
    ,DAYS=days
    ,RESULT=internal2
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

FRDATE=*internal1*

Specifies the internal date to be subtracted from.

Type: RX-type label, (Rn), internal packed (4 bytes)

DAYS=*days*

Specifies the number of days to be subtracted.

Type: RX-type label, (Rn), internal packed (4 bytes), or a constant (quotes)

RESULT=*internal2*

Specifies the field to contain the resulting internal date.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE SUB_DAYS Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  SUB_DAYS,ANCHOR=ANCHOR,PARM=DATEPARM,          X
      FRDATE=TODAY,DAYS=FWORD,RESULT=CLNDATE,           X
      FAIL=ABEND004,ERROR=ABEND004
```

TLMDATE SUB_YEARS Macro—Subtract Years From Date

Use the SUB_YEARS form of the TLMDATE macro to subtract a number of years from a date. This function subtracts YEARS from FRDATE and places the resulting date into RESULT.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE SUB_YEARS
   ,FRDATE=internal1
   ,YEARS=years
   ,RESULT=internal2
   ,ANCHOR=anchor
   ,PARM=plist
   [,NORMAL=branch]
   [,ERROR=branch]
   [,FAIL=branch]
```

FRDATE=*internal1*

Specifies the internal date to be subtracted from.

Type: RX-type label, (Rn), internal packed (4 bytes)

YEARS=*years*

Specifies the number of years to be subtracted from FRDATE.

Type: RX-type label, (Rn), packed (4 bytes) or a constant (quotes)

RESULT=*internal2*

Specifies the field to contain the resulting internal date.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE SUB_YEARS

```
                                                    Pos. 72
                                                    |
TLMDATE  SUB_YEARS,ANCHOR=ANCHOR,PARM=DATEPARM,     X
      FRDATE=CURRENT,YEARS=FWORD,RESULT=NYEAR,      X
      FAIL=ABEND004,ERROR=ABEND004
```

TLMDATE TEST_FORMAT Macro—Test External Date Format

Use the TEST_FORMAT form of the TLMDATE macro to test an external date format for CA TLMS compatibility.

This macro has the following format:

```
label TLMDATE TEST_FORMAT
    ,FMT=datefmt
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
```

FMT=*datefmt*

Specifies a 10-byte field containing a date pattern.

Type: RX-type label, (Rn), or constant (quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

Example: TLMDATE TEST_FORMAT

```
                                                        Pos. 72
                                                        |
TLMDATE  TEST_FORMAT,ANCHOR=ANCHOR,FMT=PERFDATE,        X
      PARM=DATEPARM,ERROR=ABEND004,FAIL=ABEND004
```

TLMDATE TO_EXTERNAL Macro—Convert Internal Date

Use the TO_EXTERNAL form of the TLMDATE macro to convert from an internal date to a CA TLMS external date.

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE TO_EXTERNAL
    ,TODATE=external
    ,FRDATE=internal
    ,FMT=format
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,ZEROS=branch]
    [,JDATE=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,STATS=branch]
    [,AGE=branch]
    [,MSG=branch]
```

TODATE

Specifies the field to receive the translated date.

Type: RX-type label, external character date (10 bytes)

FRDATE

Specifies the date to be converted.

Type: RX-type label, (Rn), internal packed (4 bytes)

FMT

Specifies a 10-byte field containing a date pattern.

Type: RX-type label, (Rn), or constant (quotes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If neither is specified, the default is to NORMAL.

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label

Example: TLMDATE TO_EXTERNAL Macro

```
                                                        Pos. 72
                                                        |
TLMDATE   TO_EXTERNAL,TODATE=PERFDATE,FMT=WORKDATE,     X
      FRDATE=(R5),ANCHOR=DAANCHOR,PARM=DATEPARM
```

TLMDATE TO_EXPLODED Macro—Convert Internal Date

Use the TO_EXPLODED form of the TLMDATE macro to convert a CA TLMS internal date to an exploded format. The extracted date related data is stored in the area generated by the TLMDATE EXPLODED_DATA macro. See EXPLODED_DATA for a list of information that can be extracted.

This macro has the following format:

```
label TLMDATE TO_EXPLODED
    ,TODATE=exploded
    ,FRDATE=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,ZEROS=branch]
    [,JDATE=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,STATS=branch]
    [,AGE=branch]
    [,MSG=branch]
```

TODATE

Specifies a field defined using the EXPLODED_DATA form of the TLMDATE macro.

Type: RX-type label, (Rn)

FRDATE

Specifies the internal date to be converted.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If neither is specified, the default is to NORMAL.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label

Example: TLMDATE TO_EXPLODED Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  TO_EXPLODED,ANCHOR=ANCHOR,PARM=DATEPARM,       X
      TODATE=EXPLODED,FRDATE=TMEXPDT
```

TLMDATE TO_HDR1 Macro—Convert From Internal Date

Use the TO_HDR1 form of the TLMDATE macro to convert a date in CA TLMS internal format to a date in HDR1 format. This function translates FRDATE (CA TLMS internal format) to TODATE (HDR1 format).

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE TO_HDR1
    ,TODATE=hdr1date
    ,FRDATE=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,ZEROS=branch]
    [,JDATE=branch]
    [,PERM=branch]
```

TODATE=*hdr1date*

Specifies the field to receive the converted date.

Type: RX-type label, HDR1 format, 6 bytes

FRDATE=*internal*

Specifies the date to be converted.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If neither is specified, the default is to NORMAL.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branch is performed only if the return from CTSDATE is NORMAL. It represents dates that are CA TLMS keywords.

PERM=branch

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

Example: TLMDATE TO_HDR1 Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  TO_HDR1,FRDATE=XEXPDT,TODATE=WORK1,            X
      ANCHOR=ANCHOR,PARM=DATEPARM
```

TLMDATE TO_JFCB Macro—Convert From Internal Date

Use the TO_JFCB form of the TLMDATE macro to convert a date in CA TLMS internal format to a date in JFCB format. This function translates FRDATE (CA TLMS internal format) to TODATE (JFCB format).

This function uses registers 0, 1, 14 and 15. These registers cannot be used to pass information to the macro.

This macro has the following format:

```
label TLMDATE TO_JFCB
    ,TODATE=jfcbdate
    ,FRDATE=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,ZEROS=branch]
    [,JDATE=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,STATS=branch]
    [,AGE=branch]
    [,MSG=branch]
```

TODATE=*jfcbdate*

Specifies the field to receive the converted date.

Type: RX-type label, JFCB format, hexadecimal length 3

FRDATE=*internal*

Specifies the date to be converted.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If neither is specified, the default is to NORMAL..

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label

Example: TLMDATE TO_JFCB Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  TO_JFCB,FRDATE=TMCRTDT,TODATE=JFCBDT,          X
      ANCHOR=ANCHOR,PARM=DATEPARM
```

TLMDATE TO_PREFERRED Macro—Convert From Internal Date

Use the TO_PREFERRED form of the TLMDATE macro to convert from an internal date to the CA TLMS Preferred date.

This macro has the following format:

```
label TLMDATE TO_PREFERRED
    TODATE=external
    FRDATE=internal
    ,ANCHOR=anchor
    ,PARM=plist
    [,NORMAL=branch]
    [,ERROR=branch]
    [,FAIL=branch]
    [,KEYWORD=branch]
    [,USER=branch]
    [,CATLG=branch]
    [,CATLGDD=branch]
    [,CYCLE=branch]
    [,FOREIGN=branch]
    [,LDATE=branch]
    [,PERM=branch]
    [,STATS=branch]
    [,AGE=branch]
    [,MSG=branch]
```

TODATE

Specifies the field to receive the translated date.

Type: RX-type label, external date (10 bytes)

FRDATE

Specifies the internal date to be converted.

Type: RX-type label, (Rn), internal packed (4 bytes)

ANCHOR=*anchor*

Points to the label specified by the ANCHOR form of the TLMDATE macro.

Type: RX-type label or (Rn)

PARM=*plist*

Points to the label specified by the PARM form of the TLMDATE macro.

Type: RX-type label or (Rn)

NORMAL=*branch*

(Optional). Specifies the instruction to receive control on return from the CTSDATE module when no errors occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

ERROR=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a date or range error occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

FAIL=*branch*

(Optional) Specifies the instruction to receive control on return from the CTSDATE module when a critical error has occurred. If not defined, the next instruction after the macro is executed.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. If they are omitted, NORMAL is assumed.

JDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is a Julian date.

Type: RX-type label

ZEROS=*branch*

(Optional) Specifies the label to get control if the resulting date is equal to zeros.

Type: RX-type label

The following branches are performed only if the return from CTSDATE is NORMAL. They represent dates that are CA TLMS keywords. If the value KEYWORD= is coded, all branches default to this value. If KEYWORD= is not coded, the default is to NORMAL.

AGE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword AGE/ddd.

Type: RX-type label

CATLG=*branch*

(Optional) Points to the label to get control if the resulting date is the CA TLMS keyword CATLG or CATALOG.

Type: RX-type label

CATLGDD=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword CATLG/ddd.

Type: RX-type label

CYCLE=*branch*

(Optional) Specifies to the label to get control if the resulting date is the CA TLMS keyword CYCLE/ccc.

Type: RX-type label

FOREIGN=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword FOREIGN.

Type: RX-type label

KEYWORD=*branch*

(Optional) Specifies the label to get control if the resulting date is a CA TLMS keyword (associated branch for that keyword was not coded).

Type: RX-type label

LDATE=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword LDATE/ddd.

Type: RX-type label

MSG=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword MSG/mmm.

Type: RX-type label

PERM=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword PERM or PERMANENT.

Type: RX-type label

STATS=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword STATS/nnn.

Type: RX-type label

USER=*branch*

(Optional) Specifies the label to get control if the resulting date is the CA TLMS keyword USER/nnn.

Type: RX-type label

Example TLMDATE TO_PREFERRED Macro

```
                                                        Pos. 72
                                                        |
TLMDATE  TO_PREFERRED,TODATE=C1IDATE,FRDATE=TMHDDATE,   X
      ANCHOR=ANCHOR,PARM=DATEPARM
```

Appendix B: Volume Master File Structure

The Volume Master File is a BDAM file which can be accessed through programs TLMSVMRW and TLMSVMIO. The file contains multiple record types, with related records chained together. Volume records can be accessed by an internal index structure.

The Volume Master File consists of a 500-byte record for each volume in the tape library. The name, number, status and characteristics of each data set in the tape library are maintained in this file. The following records comprise the VMF:

C/CONTROL

The control record (type C) is the first in the file. It contains such information as the company name, Auxiliary control statement, Message control statement (YES/NO), Scratch control statement (YES/NO), information about the CPUs which access it, and pointers to other areas of the VMF.

S/SEGMENT

One or more segment records (type S) follow the control record. Each of these define multiple volume serial number ranges of the VMF. Each definition contains the first VOLSER, the number of VOLSERs, and a relative block pointer to the first volume base record (type B) for the range. Segment records are used to calculate the relative block of a volume serial number. There is at least one segment record, and there can be as many as necessary to define all base record segments.

B/BASE

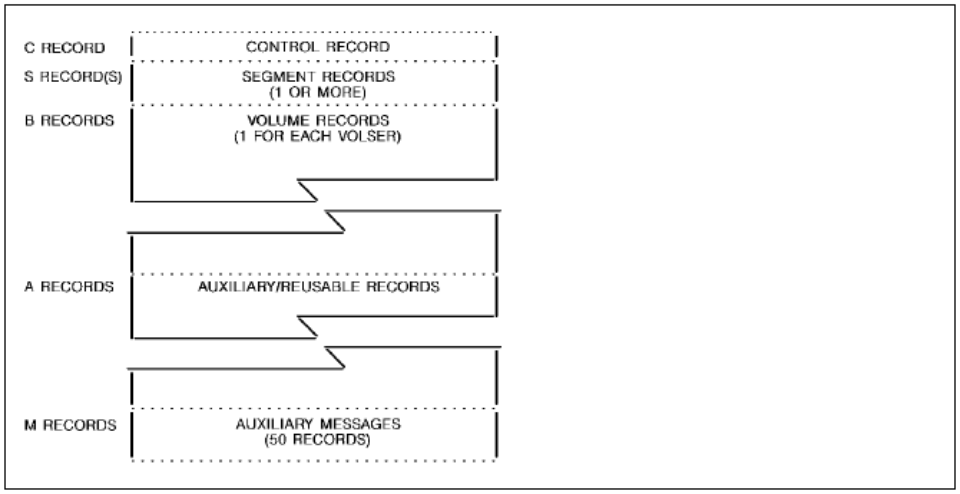
The volume base records (type B) follow the segment records. There is one base record for each volume defined in the VMF. This record contains information about the VOLSER and the first data set residing on it. Each record also contains a pointer to the *next* and *last* data set records (type D) which exist on this volume. If the volume is the first of a chain, it contains information to point to the other volumes in the chain. A base record can hold information for the first six volumes in a multivolume file. After this, a multivolume record is needed.

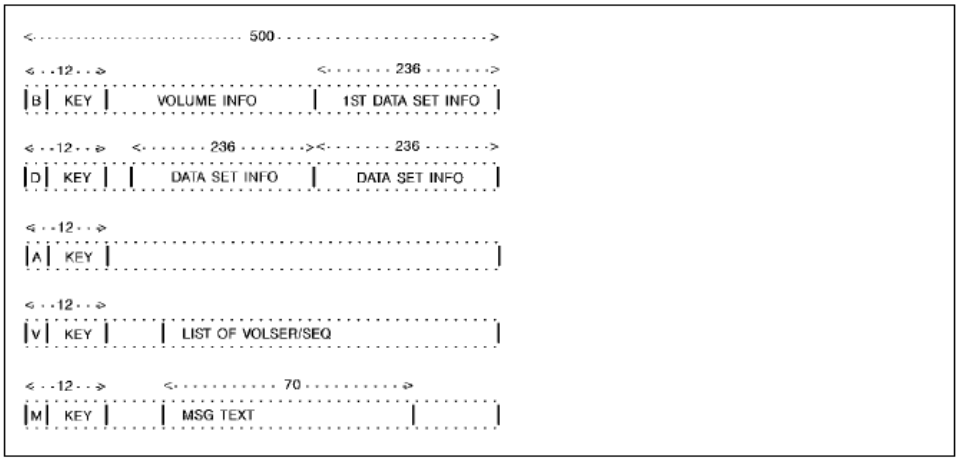
A/AUXILIARY/REUSABLE

These records are chained together and pointed to by an entry in the control record. When CA TLMS needs to write a multi-data set record (type D) or a multivolume record (type V), an auxiliary record is used. It is first removed from the auxiliary chain or from the reuse chain, and then rewritten as a type D or V, and then re-chained to the appropriate volume record. If a type D or type V record is to be deleted, it is rewritten as an auxiliary record and placed on the reuse chain.

M/AUXILIARY MESSAGES

Auxiliary message records are optional. If they are requested (MSG=YES|NO is specified when initializing VMF), they exist as the last 50 records in the VMF.





Appendix C: Database Record Definitions

CA TLMS provides macros in CAI.CTAPMAC which can be used to produce listings which describe the RMF, VMF and Transaction database records. These macros are:

TLMRMF

Produces a listing of the Retention Master File record definitions

TLMVMF

Produces a listing of the Volume Master File record definitions

TLMTRN

Produces a listing of the Transaction record definitions

Parameter values are supplied which create unique data labels, generate a DSECT, equate pre-Release 5.4 data labels to Release 12.6 Second Edition labels, print comments and determine what type of record is described. If you have code using pre-Release 5.4 data labels, you should change it to use Release 12.6 Second Edition labels, because future releases may not contain the old names.

The TLMRMF, TLMVMF and TLMTRN macros are similar. The following reproduction of the front of the TLMVMF macro is used as an example of these record definition macros.

```
*********************************************************************
*                  VOLUME MASTER FILE RECORDS                       *
*********************************************************************
* THIS MACRO GENERATES ALL VMF RECORD DESCRIPTIONS. THE FIRST TWO *
* CHARACTERS OF THE LABEL ARE USED TO MAKE RECORD LABELS UNIQUE.    *
* THE PRE-TLMS 5.4 LABELS MAY ALSO BE EQUATED TO ONE DEFINITION OF  *
* OF THE RECORDS.                                                   *
*                                                                   *
*   PARMS:                                                          *
*         DEFINE = DEFINE VALUES FOR FIELDS. DEFAULT=YES            *
*                  YES - USE 'DC' TO DEFINE DEFAULT VALUE FOR FIELD.*
*                  NO  - USE 'DS' DO NOT DEFINE FIELD. THIS VALUE   *
*                        FORCED WHEN 'DSECT=YES'                    *
*         DSECT  = GENERATE A DSECT FOR THE RECORD. DEFAULT=NO      *
*                  YES - GENERATE '&P'.'&TYPE'.'SCT' DSECT          *
*                  NO  - DO NOT GENERATE DSECT                      *
*         DS1    = GENERATE FIELDS FOR DATASET CELL 1.              *
*                  YES    - GENERATE WITH SAME PREFIX               *
*                  'XX'   - GENERATE WITH 'XX' AS PREFIX            *
*                  NO     - DO NOT GENERATE DETAIL FIELDS           *
*         DS2    = GENERATE FIELDS FOR DATASET CELL 2. (MDS ONLY)   *
*                  YES    - GENERATE WITH SAME PREFIX               *
*                  'XX'   - GENERATE WITH 'XX' AS PREFIX            *
*                  NO     - DO NOT GENERATE DETAIL FIELDS           *
*         EQUOLD = EQUATE OLD LABELS TO NEW LABELS. DEFAULT=NO      *
*                  YES    - OLD FIELDS ARE DEFINED AND POSITIONED   *
*                  NOLIST - SAME AS YES BUT FIELDS NOT PRINTED.     *
*                  NO     - OLD FIELDS ARE NOT DEFINED OR LISTED.   *
*         LAB    = LABEL/PREFIX. 1-8 CHAR. REQUIRED.                *
*                  1ST 1-2 CHAR USED AS PREFIX FOR GENERATED LABELS.*
*                                                                   *
*         LIST   = PRINT RECORD COMMENTS. DEFAULT=NO                *
*                                                                   *
*         TYPE   = TYPE OF RECORD GENERATED.  DEFAULT=BASE          *
*                  BASE = VOLUME RECORD                             *
*                  BSE  = VOLUME RECORD                             *
*                  MDS  = MULTIPLE DATASET RECORD                   *
*                  CTL  = CONTROL RECORD FOR VMF                    *
*                  MVOL = MULTIPLE VOLUME RECORD                    *
*                  MVL  = MULTIPLE VOLUME RECORD                    *
*                  SEG  = VOLSER SEGMENT DEFINITION RECORD          *
*                  MSG  = MESSAGE RECORD                            *
*                  AUX  = AUXILLARY RECORD                          *
*                  DSC  = DATASET CELL FOR A VMF RECORD             *
*                  DSN  = DATASET CELL FOR A VMF RECORD             *
*********************************************************************
```

Appendix D: Expiration Date JCL Keywords

CA TLMS provides the following expiration and retention for tape data sets methods:

- Specific criteria or unique Julian date keywords specified in the JCL LABEL parameter
- Retention Master File control statements

The following table describes the specifications that can be used in the JCL LABEL parameter for retention types that use expiration date keywords.

The keyword, external, internal, JCL and retention values are summarized below. For more information, see the chapters, “Understanding Tape Retention” and "Reports".

Keyword	External	Internal	JCL	Retention
AGE	AGE/ddd	9992ddd	992dd	5
CATLG	CATLG	9990000	99000 or 90000	1
CATLG/ddd	CATLG/ddd	9990ddd(*)	991dd	3
CYCLE	CYCLE/cc	9999ccc	990cc	4
FOREIGN	FOREIGN	9998000	98000	
JDATE	yyyy/ddd	ccyyddd	yyyy/ddd	9
LDATE	LDATE/ddd	9998ddd	98ddd	8
MSG	MSG/mmm	9991mmm	970mm	
OJDATE	yyddd	ccyyddd	yyddd	
PERM	PERM	9999999	99365 or 99366	7
STATS	STATS/sss	9989sss	none	7
USER	USER/uuu	9988uuu	88uuu	7
ZEROS	ZEROS	0000000	none	

(*) Catalog Control is uniquely translated as century 99, year 90, day 000.

External Keywords

The following exteranal expiration date keywords are available:

AGE/*ddd*

Retain data set for *ddd* days after it was created.

JCL: 9992*dd*

CATLG

Retain while data set is cataloged to the operating system.

JCL: 99000

CATLG/*ddd*

Retain *ddd* days, then retain while data set is cataloged to the operating system.

JCL: 991*dd*

CYCLE/*ccc*

Retain *ccc* cycles.

Range: 001 to 364

JCL: 990cc

FOREIGN

Tape is not a CA TLMS controlled tape, even if its volser is in the VMF. Transaction is created but the VMF is not updated.

JCL: 98000

JDATE

Julian date format.

Range: 1960/001 to 2155/366

JCL: yyyy/ddd

LDATE/*ddd*

Retain ddd days after date on which tape was last used.

JCL: 98ddd

MSG

Dismount tape message.

JCL: 970mm

OJDATE

Old Julian date format.

Range: 60001 to 59365

(60001 = 1960/001)

(59365 = 1959/365)

JCL: yyddd

PERM PERMANENT

Retain data set permanently.

JCL: 99365,99366

STATS/*sss*

Status of held tape where sss is the reason code indicating why the tape is being held. This keyword has no JCL equivalent. It is set by programs or is entered through the keyword format STAT/sss to apply permanent retention other than 99365 to an unknown situation, such as a broken chain.

USER/*uuu*

This keyword allows you to create your own keywords for processing.

CA TLMS treats this as PERMANENT.

JCL: 88uuu

ZEROS

A date with an internal value of zeros is always displayed as blanks. However, the date can be entered as any number of 0s, blanks, nulls, or the word ZEROS.

Appendix E: Integration with CA OPS/MVS EMA

This section contains the following topics:

Overview

CA TLMS provides seamless integration with CA OPS/MVS by automatically communicating both active status events and heart beat events to CA OPS/MVS. The enabling technology for this is through a generic event API call that CA OPS/MVS provides the other mainframe products so that they can communicate events to CA OPS/MVS.

You do not need to do anything for CA TLMS to enable this event communication interface to CA OPS/MVS. If CA TLMS and CA OPS/MVS are active in the same z/OS image, CA TLMS automatically communicates these automation events to CA OPS/MVS.

By generating active status events CA TLMS and other CA products are able to communicate to CA OPS/MVS's System State Manager (SSM) component when they are starting, up, stopping or down.

SSM is a built-in feature that uses an internal relational data framework to proactively monitor and manage started tasks, online applications, subsystems, JES initiators, and other z/OS resources including your CA mainframe products. SSM compares the current state of online systems, hardware devices, and the other resources with their desired state, and then automatically makes the necessary corrections when a resource is not in its desired state. This provides proactive and reactive state management of critical resources.

Before the CA OPS/MVS interface existed, CA OPS/MVS could automate active status events for your CA products; however this typically required monitoring unique messages for each CA product. With this interface, CA OPS/MVS can capture these events for any of your CA products with a single automation event rule.

With the heart beat event, CA TLMS can communicate a normal, warning, or problem overall health status and reasoning to CA OPS/MVS on a regular interval. Once CA TLMS begins generating heart beat events for CA OPS/MVS, CA OPS/MVS can also react to the lack of a heart beat event from CA TLMS, treating this as an indication that there is either a potential problem with CA TLMS, or there is a larger system-level problem that is taking place.

Ensure that CA OPS/MVS Is Enabled for Capturing These Events

To ensure that this CA OPS/MVS interface is active, make sure the CA OPS/MVS parameter APIACTIVE is set to its default of ON. This allows CA OPS/MVS to acknowledge and process the events generated by CA TLMS and other CA products through this interface.

CA TLMS Active State Events

CA TLMS provides a direct interface to the CA OPS/MVS System State Manager (SSM) application to notify CA OPS/MVS of the current operating state of the given CA TLMS address space. The CA OPS/MVS SSM application can use this information to automatically control the operation of the CA TLMS address space, as well as any other address space that is dependent upon the CA TLMS address space being active. For more information on using CA OPS/MVS SSM see the CA OPS/MVS User Guide.

The CA TLMS product active state is presented to CA OPS/MVS and can be processed by the following rule:

```
)API CASTATE
```

The available OPS/REXX variables for CA TLMS product state management are:

OPS/REXX Variable	Value
API.APPLICATION	CA TLMS
API.VERSION	Current release
API.LEVEL	00000
API.EVENTID	CASTATE
API.MSGID	CASTATE
API.TEXT	State of CA TLMS

The API.TEXT variable has the following format:

State of *appl_id* is *current_state*'

appl_id

Specifies the same value as the API.APPLICATION variable

current_state

STARTING

Indicates that CA TLMS is initializing

UP

Indicates that CA TLMS is active

STOPPING

Indicates that CA TLMS is terminating

DOWN

Indicates that CA TLMS is exiting the system

For more information on how to use the CASTATE API, see the member SSMCAAPI of opsmvsHLQ.STATEMAN.RULES.

CA TLMS Heartbeat Events

CA TLMS provides a continuous heartbeat event directly to CA OPS/MVS. CA OPS/MVS can use this information in several ways to determine the operational health of the CA TLMS product.

CA TLMS issues a heartbeat update every nnnn seconds that notifies CA OPS/MVS of the current operational health of the CA TLMS product.

If CA TLMS detects a health state change, it immediately generates a heartbeat update without waiting for the nnnn second heartbeat interval to expire. In this way, CA TLMS provides CA OPS/MVS with a constant operational health state view of the CA TLMS product.

CA OPS/MVS can also react to the lack of a heartbeat update from CA TLMS and an indication that there is either a potential problem with CA TLMS, or there is a larger system level problem that is taking place.

The CA TLMS product heartbeat event is presented to CA OPS/MVS and can be processed by the following rule:

```
)API CAHEARTBT
```

The available OPS/REXX variables for CA TLMS state management are:

OPS/REXX Variable	Value
API.APPLICATION	CA TLMS
API.VERSION	Current release
API.LEVEL	00000
API.EVENTID	CAHEARTBT
API.MSGID	CAHEARTBT
API.TEXT	State of CA TLMS

The API.TEXT variable has the following format:

appl_id Status: *heartbeat*_state Reason: reason_text

appl_id

Specifies the value of the API.APPLICATION variable.

heartbeat_state

Heart_beat_state can be one of the following:

NORMAL

Indicates that CA TLMS is operating normally, without any detected problems.

WARNING

PROBLEM

reason_text

reason_text explains the problem as reported by the event API call.

For information on how you use the CAHEARTBT API, see members APIHRTB1, APIHRTB2, and APIHRTB3 of opsmvsHLQ.SAMPLE.RULES.

Appendix F: CA TLMS Health Checks

This section contains the following topics:

Parameter Overrides for CA TLMS Health Checks

The IBM Health Checker for z/OS lets you override selected default parameters. To override parameters, you specify the desired defaults on the POLICY statement in the HZSPRMxx member of parmlib. This is useful for changing values such as INTERVAL to a value more appropriate for your installation. For a complete list of parameters that can be overridden see “Managing Checks” in IBM Health Checker for z/OS User Guide.

You can write individual checks to support parameter overrides using the PARM( ) parameter on the POLICY statement. You can also use the MODIFY command to override the parameters and pass the desired parameters to the IBM Health Checker for z/OS started task.

```
F hzsproc,UPDATE,CHECK(check_owner,check_name),PARM='chkparm'
```

An example of an override follows:

```
F HZS,UPDATE,CHECK(CA_TLMS,TLMS_AUX_CUSHION_WARNING),

              PARM=' CUSHIONTHRESH(60)'
```

If a CA TLMS check supports the override of a parameter, it is documented in the Parameters Accepted section in each check that follows. The default interval and the exception interval are provided in the Description section for each CA TLMS check.

TLMS_AUX_CUSHION_CRITICAL

Description

This check monitors the availability of AUX records in the CA TLMS VMF and triggers an exception when the number of AUX records used exceeds a critical percentage. This check is a secondary check that triggers if the AUX record shortage warned about in check TLMS_AUX_CUSHION_WARNING has not been addressed. The default interval for this check is every 24 hours and the exception interval is every 15 minutes.

Best Practice

CA TLMS uses AUX (Auxiliary) records to track secondary files and multi-volume chains. If you run out of AUX records you cannot track multi-volume or multi-file tape data sets. As a CA TLMS best practice, monitor the count “TOTAL AUXILIARY RECORDS WRITTEN” in the output from the CATVMFB daily VMF backup utility. This represents the number of AUX records available for chaining. When this number drops suddenly and significantly you should determine why. It can indicate a problem to investigate.

Parameters accepted

This check accepts one parameter, CUSHIONTHRESH(nn). This parameter specifies the percentage of AUX records that was used. Calculated values above this percentage cause the check to raise an exception. This parameter must be an integer in the range of 1-99. The default is CUSHIONTHRESH(75).

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see CATVMFB - Back Up the Volume Master File (see page 216).

Message

TLMSH0031E The CA TLMS VMF is running out of AUX records.

More information is available in the Message Reference Guide.

TLMS_AUX_CUSHION_WARNING

Description

This check monitors the availability of AUX records in the CA TLMS VMF and triggers an exception when the number of AUX records used exceeds a warning level. The default interval for this check is every 24 hours and the exception interval is every 15 minutes.

Best Practice

CA TLMS uses AUX (Auxiliary) records to track secondary files and multi-volume chains. If you run out of AUX records you cannot track multi-volume or multi-file tape data sets. As a best practice, monitor the TOTAL AUXILIARY RECORDS WRITTEN count in the output from the CATVMFB daily VMF backup utility. This represents the number of AUX records available for chaining. When this number drops suddenly and significantly you should determine why. It can indicate a problem to investigate.

Parameters accepted

This check accepts the parameter CUSHIONTHRESH(nn). This parameter specifies the percentage of AUX records used that raise an exception for this check. This parameter must be an integer in the range of 1-99. The default is CUSHIONTHRESH(50).

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see CATVMFB - Back Up the Volume Master File (see page 216).

Message

TLMSH0041E The CA TLMS VMF is running low on AUX records. More information is available in the Message Reference Guide.

TLMS_OPTION_NOTLMS

Description

This check examines the setting of the NOTLMS option and creates an exception when NOTLMS=CONT. The NOTLMS option determines the action of its tape open intercepts when the CA TLMS address space is not active. These intercepts take control during tape open processing to control access to the tape file and capture information about the tape file. If the CA TLMS address space is not active it cannot direct the tape intercepts in these functions.

You can set the NOTLMS option to NOTLMS=ABEND or NOTLMS=CONT. When NOTLMS=ABEND, the tape intercepts abnormally terminate jobs opening a tape file for input and reject tapes being opened for output.

When NOTLMS=CONT, the tape intercepts allow the tape open to continue and no information is captured about the file.

The default interval for this check is to run once when the CTS address space starts up and the exception interval is your system default.

Best Practice

For CA TLMS best practices, NOTLMS should be set to ABEND. This lets CA TLMS control and record tape activity. Use NOTLMS=CONT only in special situations where it is necessary to process a limited number of tape files without CA TLMS control.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options (see page 33).

Message

TLMSH0061E The CA TLMS option NOTLMS is set to CONT, which allows tape jobs to continue to run even though CA TLMS is not available to update the VMF. More information is available in the Message Reference *Guide*.

TLMS_OPTION_PROTECT

Description

This check examines the setting of the PROTECT option and creates an exception when PROTECT=SELECT.

The default interval for this check is to run once when the CTS address space starts up and the exception interval is your system default.

Best Practice

For CA TLMS best practices, specify PROTECT=ALL. This ensures that the tape is removed from scratch status and the volume is protected from being overwritten.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options (see page 33).

Message

TLMSH0071E The CA TLMS option PROTECT is set to SELECT. If a system crash occurs, volumes that were actively being written to are not protected from reuse by another application. More information is available in the Message Reference Guide.

TLMS_OPTION_RECOVRY

Description

This check examines the setting of the RECOVRY option and creates an exception if RECOVRY=NONE or RECOVRY=SMF is specified.

The default interval for this check is to run once when the CTS address space starts up and the exception interval is your system default.

Best Practice

For CA TLMS best practices, specify RECOVRY=ALTLOG. This ensures that transactions are recorded in the ALOG data set and provide the fastest and most reliable method of recovery

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options in this guide.

Message

TLMSH0081E The CA TLMS option RECOVRY is set to NONE. If recovery is required, transactions recorded since the last full VMF backup will be lost.

TLMSH00821E The CA TLMS option RECOVRY is set to SMF. If recovery is required, transactions recorded since the last full VMF backup will be more difficult to recover than from the ALTLOG.

More information is available in the Message Reference Guide.

TLMS_OPTION_SECOPN

Description

This check examines the setting of the SECOPN option and creates an exception if SECOPN=NO is specified.

The default interval for this check is to run once when the CTS address space starts up and the exception interval is your system default.

Best Practice

For CA TLMS best practices, define rules for your tape data sets to CA Top Secret, CA ACF2, or IBM RACF. Also define rules for CA tape resources “CATAPE”,”CACMD” and “PANEL”. Then set the CA TLMS global security option to SECURE=YES and set the open security option to SECOPN=YES. This allows CA TLMS to make the necessary security calls during tape open.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options (see page 33) and Security System Interface (see page 305).

Message

TLMSH0091E The CA TLMS option SECOPN is set to NO. Your tapes are not protected from unauthorized access. More information is available in the Message Reference Guide.

TLMS_OPTION_SECURE

Description

This check examines the setting of the SECURE option and creates an exception if SECURE=NO is specified.

The default interval for this check is to run once when the CTS address space starts up and the exception interval is your system default.

Best Practice

For CA TLMS best practices, define rules for your tape data sets to CA Top Secret, CA ACF2, or IBM RACF. Also define rules for CA tape resources “CATAPE”,”CACMD” and “PANEL”. Then set the CA TLMS global security option to SECURE=YES This allows CA TLMS to make the necessary security calls for the CA TLMS security options you enabled.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options (see page 33) and Security System Interface (see page 305).

Message

TLMSH0101E The CA TLMS option SECURE is set to NO. CA TLMS does not issue any security calls to tape open, close, or access to CA TLMS through ISPF. More information is available in the Message Reference Guide.

TLMS_QUEUE_ACTIVE

Description

This check monitors the status of the CA TLMS transaction queue and creates an exception if queue processing is stopped.

The default interval for this check is every 15 minutes and the exception interval is once per minute.

Best Practice

For best practices, keep CA TLMS active at all times. If CA TLMS is not active, tape files could be over written and the content of tape volumes are unknown since tape data set information is not captured.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options (see page 33) in this guide.

Message

TLMSH0011E The CA TLMS queue is inactive and TLMS cannot process requests. More information is available in the Message Reference Guide.

TLMS_VMF_ALOG_SEPERATION

Description

This check determines whether the VMF and ALOG reside on the same disk volume and creates an exception if they do. An exception is not created if the CA TLMS recovery option is set to RECOVRY=NONE or RECOVRY=SMF.

The default interval for this check is to run once when the CTS address space starts up and the exception interval is your system default.

Best Practice

For CA TLMS best practices, allocate the VMF and the ALOG on different non-SMS disk volumes. This ensures that the VMF can be recovered in the event the disk volume fails. The recovery procedure for the VMF (CATVMFR) creates a recovered VMF by reading the last full VMF backup and applying all the transaction that occurred after the backup. The transactions are recorded in the ALOG and its backups.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see System Options (see page 33).

Message

TLMSH0021E The CA TLMS VMF and ALOG files should not be on the same disk volume. More information is available in the Message Reference Guide.

TLMS_VMF_UPDATE_NOT_POSSIBLE

Description

This check monitors CA TLMS updates to the VMF to see if any volume chaining or other errors occurred. The default interval for this check is to run every hour. If an exception occurs the check clears the count of update failures.

Best Practice

CA TLMS uses a chaining technique to link volume base records that are created when multivolume data sets are created. These chains are important for proper functioning of CA TLMS. The volume chaining verification procedure CATVCVS is provided and should be run regularly as a best practice. This check provides an early warning that a volume chaining error was encountered in CA TLMS processing.

Parameters accepted

None.

Debug Support

Yes

Verbose Support

Yes

Reference

For more information, see Maintaining the Volume Master File (see page 293).

Message

TLMSH0051E There have been &nnn update failures to the VMF within the last hour. CA TLMS could not properly update the VMF. For more information, see the Message Reference Guide.

Index

D

E

I

J

L

M

O

P

R

S

T

U

V