CA Cloud Storage for System z

z/OS Configuration Guide

Release 1.1.00

CA Technologies Product References

The CA Cloud Storage for System z guides refer to the following CA products and components:

- CA 1® Tape Management (CA 1)
- CA Allocate™ DASD Space and Placement (CA Allocate)
- CA Earl® (CA Earl)
- CA Chorus Software Manager™ (CA CSM)
- CA MIM™ Resource Sharing (CA MIM)
- CA Tape Encryption
- CA TLMS® Tape Management (CA TLMS)
- CA Vtape™ Virtual Tape System (CA Vtape)

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 1: About this Guide

This section contains the following topics:

Audience

Users of this guide should be experienced mainframe technicians with knowledge of their mainframe tape systems, tape related software, and security configuration.

Conventions

The following conventions are used throughout this guide to document features, functions, and other aspects of the system:

- Variable text, most commonly used for data set names and console commands, is entered in italics.

  For example, VVE_SCRATCH=*volser* where *volser* is the Virtual Volume VOLSER.

- Commands are entered in uppercase and lower-case. The uppercase portion is the minimum number of characters that must be entered for CA Vtape to recognize the command. The lower-case portion is provided for clarity.

- Features, functions, and components of CA Cloud Storage for System z are capitalized. These include, for example: Virtual Volumes and Virtual Devices.

Chapter 2: Setting up the System, Tape Management, and Third-Party Products

This chapter describes the system, tape management, and third party setup to successfully implement CA Vtape.

This section contains the following topics:

CA Cloud Storage for System z Installation

If you have not started the CA Cloud Storage for System z installation, see the *z/OS Installation Guide*.

Limited-Use CA Vtape License Requirement

New and existing CA Vtape customers require the Limited-Use CA Vtape License to implement CA Cloud Storage for System z.

How to Configure z/OS for CA Cloud Storage for System z

Important! Complete the installation instructions in the *CA Cloud Storage for System z z/OS Installation Guide,* then perform this configuration.

Follow these steps:

1. Perform the critical tasks that are documented in this section to define CA Vtape to the operating system, the tape management system, and other third-party software.
2. Review "ISPF Customization Panels (see page 43)" and use the ISPF customization panels to create JCL to define the control data sets and to create the configuration member SUTPARMS.
3. Follow the instructions in "Product Verification (see page 119)" to customize the Parmlib, start the product, and verify the product operation.
4. Review and follow the instructions in "Operational Considerations" (see page 125) to set up utility JCL to backup the control data sets, adjust the Parmlib settings and other software for your environment and needs.

The *CA Cloud Storage for System z z/OS Administration Guide* provides more information that you may need when following instructions in this guide:

- Overview of the product, its features, and its components
- CA Vtape console commands
- CA Vtape utilities

Multisystem Planning

When planning for a multisystem environment, consider the following items:

- The Global VCAT provides the mechanism for sharing and controlling Virtual Volume access between multiple CA Vtape Subsystems.
- One or more CA Vtape subsystems sharing the Global VCAT and Linux file server comprise a CA Vtape Complex.
- Implement more than one Complex by creating and sharing different Global VCATs. Typically, this is done to separate test and production data.
- For performance reasons, CA Vtape uses hardware reserves against the DASD volume on which the Global VCAT resides.
- Virtual Devices are unique to an individual system and are not shared. For example, Virtual Device F05 on system A is not the same device as Virtual Device F05 on system B because separate started tasks are emulating them. Both devices can be online and in use simultaneously on their respective systems.

- Virtual Volumes can be shared across multiple subsystems and are serialized like physical tapes.
- Concurrent read of a Virtual Volume is allowed if the reader is CA Disk. For example, if multiple CA Disk auto-restores are triggered, on one or more systems, for data sets that reside on the same Virtual Volume, CA Vtape allows the auto-restores to access that Virtual Volume simultaneously.
- The CA Vtape Parmlib supports symbolic substitution in its Parmlib member names and Parmlib attribute values. This support allows you to use a single Parmlib to support your multiple system configuration.

  Note: For more information about symbolic substitution, see the chapter “Product Verification.”

Multisystem Requirements

When setting up your multisystem environment consider the following requirements:

- Allocate the Global VCAT and BSDS on separate volumes shared by all systems.
- If you define multiple CA Vtape Complexes, allocate the Global VCAT and BSDS1 data sets pertaining to the different Complexes on separate DASD volumes shared by all systems.
- Sysplex is supported, but not required.
- Cross-system enqueue propagation is not required, but you can use the propagation to resolve contention.
- If a Tape Management System is installed, all systems in the same CA Vtape Complex should share a common Tape Management System database.

Define a Catalog Alias

CA Vtape defines and catalogs various work data sets. These data sets are discussed in detail in the chapter "SMS Setup (see page 37)." A catalog alias should be defined for these data sets to prevent them from being cataloged in the master catalog.

The default high-level qualifier for these data sets is CAVTAPE1. A different value of up to 23 characters can be used. The value used is entered into an ISPF panel and saved in the SUTPARMS configuration member.

The SUTPARMS member initializes the Global VCAT, BSDS1, and Local VCAT data sets. This member is created in the chapter "ISPF Customization Panels (see page 43)."

SVTS Reserve

CA Vtape keeps its control information in a dataspace and writes it out to the Global VCAT and BSDS data sets. To ensure the integrity of these updates, a hardware reserve is issued with QNAME=SVTS against the volume the Global VCAT resides on.

The RNAME value used is determined by setting the GlobalReserve attribute located in the Dynamic Options Section of the OSPARMS Parmlib member.

If Enhanced is coded, the RNAME value includes part of the Global VCAT and BSDS data set names. This allows multiple CA Vtape Complexes to reserve their own control data sets with a unique RNAME value. Complex A can update its own control data sets without having to wait on Complex B while it is updating its control data sets.

If COMPATIBILITY is coded, the RNAME value is GLOBAL. In a multiple CA Vtape Complex environment, a reserve issued with the generic RNAME of GLOBAL will force one complex to wait before updating its control data sets while another complex is updating its own control data sets.

Enhanced is the recommended setting.

When CA Vtape is running with GlobalReserve=Enhanced a scan is performed for the QNAME=SVTS, RNAME=GLOBAL reserve. If the Compatibility reserve is detected, the detecting CA Vtape will dynamically change to GlobalReserve=Compatibility to prevent possible control data set corruption. Corruption could occur if two Subsystems sharing the same Global VCAT and BSDS are started with different GlobalReserve settings.

You can also dynamically switch between values by updating the GlobalReserve attribute and issuing the SVT*n* REFRESH=OPTIONS console command. If you change to Compatibility, any other subsystem in the complex that is running with Enhanced automatically switches to Compatibility as soon as your change is detected. If you change to Enhanced and any other subsystem is running with Compatibility, the subsystem you just changed will switch back to Compatibility.

If you have a DASD resource serialization manager such as CA MIM or IBM Global Resource Serialization (GRS), you may want to convert QNAME=SVTS hardware reserves to SCOPE=SYSTEMS enqueues. However, in some cases this approach can lead to integrity exposures and corruption of the Global VCAT and the BSDS.

Note: The F MIM, DISPLAY INIT command can be used to determine your GDIF PROCESS=SELECT/ALLSYSTEMS operating mode value.

Important! All instances of the QNAME=SVTS hardware reserve, regardless of the RNAME value, must be managed the same way on all systems. Converting some QNAME=SVTS reserves and not others will create an integrity exposure that could result in data loss.

Also, if the QNAME=SVTS reserve is converted to an enqueue for a CA Vtape Complex that is running in multiple sysplexes and you do not use IBM's GDPS/PPRC HyperSwap, the resulting integrity exposure could cause a data loss.

The following sections describe two different implementations, the recommended implementation and the required implementation for sites using IBM's GDPS/PPRC HyperSwap product.

Recommended Implementation, IBM's GDPS/PPRC HyperSwap Product Not Used

Note: If you are not using IBM's GDPS/PPRC HyperSwap product, do not convert the QNAME=SVTS hardware reserve with CA MIM or IBM GRS to an enqueue.

- If CA MIM is in ALLSYSTEMS mode, add the following statement to the MIM MIMQNAME Parmlib member to tell CA MIM not to convert the QNAME=SVTS hardware reserves:

  ```
  SVTS GDIF=NO
  ```

- If CA MIM is in SELECT mode, you do not need any MIM MIMQNAME or GDIEXMPT Parmlib member updates. By default hardware reserves are not automatically converted.

- For IBM GRS, you do not need any GRSRNL00 Parmlib member updates. By default hardware reserves are not automatically converted.

Required Implementation, IBM's GDPS/PPRC HyperSwap Product in Use

Note: If you are using IBM's GDPS/PPRC HyperSwap or you are sure you will *never* share a Global VCAT and BSDS between more than one sysplex, you can safely have CA MIM or IBM GRS convert all instances of QNAME=SVTS hardware reserves to a SCOPE=SYSTEMS enqueue.

- If CA MIM is in ALLSYSTEMS mode, you do not need any MIMQNAME or GDIEXMPT Parmlib member updates, because all hardware reserves are automatically converted in this mode.

- If CA MIM is in SELECT mode, add the following statement to the MIMQNAME Parmlib member to convert all QNAME=SVTS hardware reserves to a SCOPE=SYSTEMS enqueue:

  ```
  SVTS GDIF=YES,
  SCOPE=RESERVES,
  EXEMPT=NO,
  ECMF=YES,
  RPTAFTER=30,
  RPTCYCLE=60
  ```

- For IBM GRS, add the following statement to the GRSRNL00 Parmlib member to convert the QNAME=SVTS hardware reserves to a SCOPE=SYSTEMS enqueue:

  ```
  RNLDEF RNL(CON)  TYPE(GENERIC)  QNAME(SVTS)
  ```

JES3 Requirements

CA Vtape can be installed in a JES3 environment, but the Virtual Devices defined for CA Vtape use cannot be placed under JES3 allocation control. To accomplish this, the following conditions must be met:

- The Virtual Devices are not defined in the JES3 Initialization Statements or INISH deck. Allocation will be handled directly by the operating system and treated like JES2 allocations.
- A unique esoteric is defined to contain the Virtual Devices.
- The Virtual Devices are defined as a device type that would not include them in a generic that includes devices controlled by JES3. For example, if 3490 devices are defined to the system and under JES3 control, the CA Vtape devices are defined as 3480 devices. If 3490 and 3480 devices are defined to the system and under JES3 control, then all of the 3490 or all of the 3480 devices are removed from JES3 control and the CA Vtape Virtual Devices are defined as the removed device type.
- High Water Mark (HWSNAME,TYPE=) and SETNAME statements included in the JES3 INISH deck do not refer to the esoteric or generic used by the Virtual Devices.
- If an existing esoteric or generic was reused for the Virtual Devices, any references to them in the JES3 Initialization Statements are removed.

Define Virtual Devices Using IBM's Hardware Configuration and Definition (HCD) Dialogs

You must define a primary and alternate set of Virtual Devices for the system to use as the primary set for virtual tape processing. If a permanent primary set failure is clearable by an IPL only, the primary set is used for virtual tape processing.

To define the Virtual Devices to the operating system, follow the Define I/O Device Data procedure in the *IBM Hardware Configuration Definition User's Guide* (http://pic.dhe.ibm.com/infocenter/zos/v1r13/index.jsp?topic=%2Fcom.ibm.zos.r13.cbdu100%2Fcbdzug90.htm) with the following changes:

1. Define the Virtual Devices as 3480s or 3490s because CA Vtape emulates these tape device types.
2. Do not define the devices with control units (CUs) or channel paths (CHPIDs) because they are software devices.
3. On the Define Device Parameters/Features screen, verify the following:
   - Autoswitch is No.
   - Dynamic is Yes.
   - Sharable is No.
   - Compact is Yes.

4. Add the newly defined Virtual Devices to new or existing esoterics for which virtual mounts are desired. We recommend that the system programmer update existing esoterics to avoid JCL changes.
5. Follow the Build a Production IODF procedure in the *IBM Hardware Configuration Definition User's Guide* to verify your changes and create a new production IODF.
6. Activate the new devices dynamically (software change) for testing.
7. Update SYS1.PARMLIB to reference the new IODF to retain the changes across IPLs.

Note: This information was developed from the *IBM z/OS V1R13.0 Hardware Configuration Definition User's Guide*. Any change to the HCD software or the *Hardware Configuration Definition User's Guide* may invalidate this information.

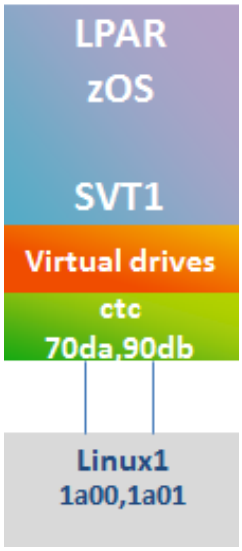
You enter the primary and alternate set of defined devices into the OSDRIVE and VTDRALT Parmlib members. For more information, see the section Initial Configuration of the Parmlib (see page 69).

Define Channel-to-Channel (CTC) Addresses

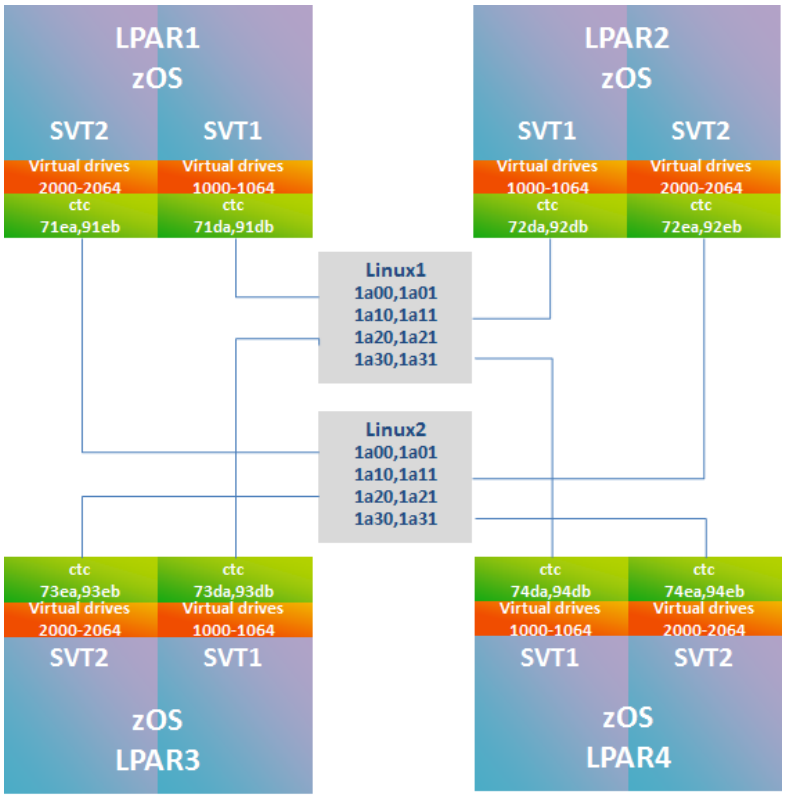
Channel to Channel adapters communicate between the CA Vtape started tasks on z/OS and the CA Cloud Storage for System z Server running on the Linux system. Using CTCs lets data be moved between hosts quickly, using the fewest general processer resources. CA Cloud Storage for System z combines multiple CTC addresses between hosts to facilitate multiplexing read and write operations for a variable number of virtual tape drives. CA Cloud Storage for System z requires two CTC addresses between each instance of CA Vtape on z/OS and Linux on System z.

Installations that have a FICON channel connected to the I/O fabric can have CTC connections set up with no additional cost. Installations no longer need to dedicate channel path (CHPID) resources to the CTC function. you can make CTC connections between channels by going through a FICON Director. The CTC adapter function is implemented logically between connecting channels and resides in the FICON channel.

The following diagram shows a single z/OS LPAR communicating with a Linux on System z host:



The following is a diagram of two z/OS LPARs communicating with a single Linux on System z host:



The IBM System z ESCON and FICON Channel-to-Channel Reference http://www-01.ibm.com/support/docview.wss?uid=isg29fafd2510f68d17285256ced0051213e&aid=1 Guide and the z/OS Hardware Configuration Definition User's Guide contain basic information, definition, and configuration information for CTCs.

Work Load Manager (WLM) Considerations

We recommend that you use SYSSTC as the WLM Service Class for the CA Vtape Started Tasks or at least a Service Class with dispatching priority above TSO and Batch workloads.

If you intend to write database logs or online transaction files for applications that run with very high WLM service classes to CA Vtape Virtual Volumes, setting CA Vtape to the same WLM Service Class would be recommended to ensure the offload occurs as quickly as possible. If CA Vtape is running in a class lower than the database or online application, activity by applications in higher classes could prevent CA Vtape from performing the offload quickly. This could result in the database of online application having to wait longer for the offload to complete.

APF Authorizations

The authorized program facility (APF) authorizes the CA Vtape SMP/E target loadlib on each LPAR where CA Vtape will be run. If you copy the SMP/E target data sets to another set of data sets (run libs), APF authorize this loadlib as well.

Logger Data Offload

The system logs its activity in a dataspace allocated at startup. This dataspace is included in dumps automatically taken by the product or when you issue the SVT*n* DUMP console command.

Note: Dataspaces are not by default included in standard console dumps. For this reason we recommend that you always use the SVT*n* DUMP console command when capturing diagnostic information about the system.

The system treats this dataspace like a wrap-around file. When the file is full, the first record is over-written.

This log can be offloaded to an IBM Log Stream and saved like SMF data for reporting and trouble shooting at a later time. We highly recommend that you define an IBM Log Stream and save this data.

The process related to the Logger dataspace is referred to as the Internal Logger. The process to offload this data to an IBM Log Stream is referred to as the External Logger.

IBM System Logger Requirements

To define a Log Stream, the IBM System Logger must be active. The IBM manual *MVS Setting up a Sysplex* provides all the necessary information required to setup the System Logger facility.

This document assumes that:

- A System Logger is active in a SYSPLEX (MONO, LOCAL, or Full SYSPLEX).
- The IXGLOGR started task is active.
- Logger policies are defined.

If these requirements are not met, the Internal Logger will write until the Logger dataspace is full, reset to the start, and overwrite the dataspace. The amount of logging data available will be limited to the size of the Logger dataspace. The size of the Logger dataspace is controlled by the LogDataspaceSize attribute in the Parmlib Startup Options Section.

To determine if the IBM System Logger is active, issue the IBM D LOGGER console command. An active logger will display as follows:

```
IXG601I   09.36.12  LOGGER DISPLAY
SYSTEM LOGGER STATUS
 SYSTEM   SYSTEM LOGGER STATUS
 ------   --------------------
 XE61     ACTIVE
```

Log Stream General Considerations

A Log Stream can be defined to the IBM System Logger as resident in the Coupling Facility or as DASDONLY. When defined as resident in the Coupling Facility a single Log Stream can be defined for all the CA Vtape Subsystems running on LPARs that share that coupling facility. When defined as DASDONLY, each Subsystem must have a different Log Stream defined for it.

We recommend that DASDONLY Log Streams be used.

The operating system will keep the External Logger data in VSAM linear data sets (LDS) for the period of time specified when the Log Stream is defined. The LDSs are managed by the operating system and are never referenced in JCL to read the logger data.

To access the data in a Log Stream, the IBM LOGR subsystem is used. You code a DD in your JCL with a DSN parameter containing the Log Stream name and a SUBSYS parameter pointing to LOGR. Data selection criteria can also be coded as part of the SUBSYS parameter. Sample JCL to access Log Stream data and write it to a sequential data set can be found in HLQ.CCUUJCL(GENLOGR).

You can customize GENLOGR to select the previous days Log Stream data and copy it to a CA Vtape set of Virtual Volumes. By scheduling this job to run daily, Log Stream data can be kept for days, weeks, or months and used for historical reporting and troubleshooting.

The IPCS utility is the reporting tool for both the Internal and External Logger. Sample JCL can be found in HLQ.CCUUJCL(IPCS).

Note: For more information about the IPCS utility, see the chapter “Troubleshooting” in the *Administration Guide*.

Define the Log Stream

The External Logger requires that you create a System Logger Log Stream. IBM's Administrative Data Utility program IXCMIAPU is used for this purpose.

See the IBM manual *MVS Setting Up a Sysplex* for all the options available for this utility.

The following sample JCL is provided in HLQ.CCUUJCL(IXCMIAPU):

```
//DEFINE   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DATA TYPE (LOGR)
    DELETE LOGSTREAM NAME(VTAPE.sysid.LOG)
    DEFINE LOGSTREAM NAME(VTAPE.sysid.LOG) DASDONLY(YES) RETPD(4)
           STG_SIZE(12288) LS_SIZE(36864) HLQ(VTAPE) AUTODELETE(YES)
```

In the sample JCL, the Log Stream name is VTAPE.*sysid*.LOG where *sysid* is the system name for the LPAR on which the logging is occurring. This name must be entered as the value of the LogStream attribute in the Parmlib Dynamic Options Section.

If you are defining multiple CA Vtape Complexes on the same LPARs, then changing the Log Stream name to VTAPE.*sysid.subsystemid*.LOG where *subsystemid* is the CA Vtape subsystem ID (SVT*n* where *n* = 1-8) is recommended.

The DASDONLY(YES)keyword instructs the System Logger to define a DASD-only Log Stream which does not require the coupling facility.

The RETPD(4) and AUTODELETE(YES) keywords instruct the System Logger to retain the log data for four days and then automatically delete it.

The STG_SIZE(12288) keyword instructs the system to create a VSAM linear data set with a size of 12,288*4096 bytes or 50 MB for staging logger records.

For DASDONLY Log Streams, staging data sets are required as part of the System Logger configuration. The System Logger automatically duplexes data to the staging data set for the system at the same time it writes the data to local storage buffers.

The LS_SIZE(36864) keyword instructs the system to automatically create VSAM linear data sets of size 36,864* 4096 bytes or 151 MBs when necessary to hold logger records as they are offloaded from the staging data set.

IBM recommends that you size the data sets as large as your installation can make them. This will minimize the number of log data sets required to represent a Log Stream. It will also minimize the number of times the System Logger must reallocate and switch to using a new log data set when an old one becomes full. Because allocating and switching to a new log data set incurs overhead, it should be done as little as possible.

By default, each Log Stream is limited to a maximum of 168 log data sets unless you define data set directory extent records in the LOGR couple data set and make it the active primary LOGR couple data set. By defining data set directory extent records, a Log Stream is no longer limited to 168 log data sets.

Without directory extent records, the maximum amount of data that can be held concurrently in the Log Stream is 151 MB * 168 or 25.4 GB based on the recommended values.

Note: The sample STG_SIZE and LS_SIZE values are based on the default setting of three for the LogDetailLevel attribute. If LogDetailLevel one or two is set instead, the sample sizes can be reduced by half.

The HLQ(VTAPE) keyword instructs the system to define the staging and logger VSAM linear data sets using the *a* high-level qualifier of VTAPE in the data set name. We recommend that you specify a value consistent with your other Log Stream data sets that includes VTAPE. If this results in a multiple level qualifier like SYS2.VTAPE, then the HLQ parameter will need to be changed to the EHLQ parameter. HLQ(VTAPE) will need to be changed to EHLQ(SYS2.VTAPE).

Tape Management Configuration

The topics in this section provide tape management considerations, sharing the tape management system database, defining exclusive tape range for virtual volumes, and subpooling.

Tape Management System Considerations

If you installed a Tape Management System, then consider these situations:

- If you want to run CA Vtape on multiple systems and to share Virtual Volumes across those systems, then share the Tape Management System Database among those systems.
- If the Virtual Volumes are defined in subpools, then define an exclusive tape VOLSER range in the Tape Management System for CA Vtape use.
- How data set stacking is supported.

Note: For more Tape Management System considerations, see “Scratch Tape Synchronization" and "Tape Management Systems.”

Tape Management System Database Share

Like physical tapes, you can share Virtual Volumes among systems. Also like physical tapes, if the systems do not share the Tape Management System (TMS) Database, the tape usage is not completely documented, the tape security can be flawed, and application JCL changes may be required to mount the shared tapes.

For example, if a tape is defined only in the TMS Database on system A and written to on system B, the mount is not documented in the Database. Because the system B Database is not controlling the usage of this tape, an unauthorized user might read or write to this tape. If the TMS is configured to disallow the mounting of tapes that are not documented in the Database, you may need to code a TMS override in the application JCL to mount the tape on system B.

Exclusive Tape Range for the Virtual Volumes

Up to 1,000,000 Virtual Volumes can be defined to a single CA Vtape Complex. Each one of these Virtual Volumes must also be defined in scratch status to the Tape Management System. Consult your tape management system manuals to determine how to define a tape range for CA Vtape to use.

You should define enough Virtual Volumes to handle the workloads redirected to Virtual Devices. The easiest way to determine this number is to run reports against your Tape Management System to determine the number of tapes currently in use, the number of data sets on those tapes, and how much data is on those tapes.

We recommend that a range of 100,000 VOLSERs be defined at minimum. Since these VOLSERs are just indexes in control data sets, they only increase the size of those control data sets. Having extra VOLSERs available is more desirable than having to take outages to resize control data sets to add VOLSERs as needed.

Note: A subset of these VOLSERs should be reserved for disaster recovery testing.

Subpooling

Subpooling is not required, but is supported in CA Vtape. If you defined subpools in your Tape Management System for the workloads to move to CA Vtape, duplicate those subpool definitions in the subpool definitions in the system Parmlib.

Note: Multiple pool definitions are not supported under JES3. You can define only one volume pool under JES3.

You can define up to eight subpools. If the workloads to move to CA Vtape currently use more than eight subpools, consolidate your Tape Management System subpool definitions. Consolidation reduces the subpool number to eight or less. Consult your Tape Management System manuals to determine how to modify your subpool definitions.

Note: For more information about defining subpools in the system Parmlib, see the chapter "The Parameter Library (PARMLIB)."

VTA08NON Job to Allocate Non-SMP/E Data Sets

Customize and submit HLQ.CCUUJCL member VTA08NON to allocate the SVTJCL and Parmlib data sets, which are non-SMP/E data sets, and copy the sample Parmlib members into the Parmlib data set.

The SVTJCL data set will contain the customized configuration JCL and a configuration member generated by the ISPF Customization Process.

Note: For more information, see the chapter “ISPF Customization Panels”.

Oracle Host Software Component (HSC) and Storage Management Component (SMC)

If CA Vtape is installed at a site that has HSC and SMC installed to manage tape mounts, you may need to modify this software to coexist with CA Vtape using one of the following methods:

- (Recommended) Code POLICY statements (see page 26).
- Install the HSC/SMC exits (see page 26).
- Update HSC/SMC parameters and allocation routines (see page 28).
- Use DFSMS or CA Allocate to perform unit replacement (see page 29).

Method One: Install POLICY Statements

The HSC/SMC software no longer supports the use of the HSC/SMC exits. POLICY statements are coded to direct which tape mounts will be intercepted by the HSC/SMC controlled tape library.

To use this method, modify your POLICY statements to not intercept those data sets that CA Vtape should intercept.

Note: For information concerning the coding of POLICY statements, see the HSC/SMC manuals for the installed release of the software.

Method Two: Install HSC/SMC Exits

The SLSUX02 exit (see page 27) influences nonspecific or scratch mounts and must be installed if Virtual Devices and silo devices share the same esoteric.

The SLSUX08 exit (see page 27) influences specific VOLSER or nonscratch mount requests. This exit must be installed if Virtual Devices and silo devices share the same esoteric and TAPEREQ statements or if tapes are ejected from the silo and must be entered back into the silo to be mounted.

POLICY statements must not be assigned to mount requests that should be intercepted by CA Vtape. If the HSC/SMC software assigns a POLICY statement to a mount request, the software does not call its exits.

SLSUX02

The CA Vtape load module VSLSUX02, checks the CA Vtape filters to determine if the mount should be routed to Virtual Devices or not.

If SLSUX02 is not currently installed, concatenate the CA Vtape SMP/E target loadlib containing the VSLSUX02 module with the HSC loadlib in the HSC started task. Consult the HSC manuals on how to activate the exit.

If the SLSUX02 exit is already in use, CA Vtape provides a driver program to combine the existing exit module and VSLSUX02. The driver source can be found in HLQ.CCUUSAMP(SLSUX02D). Follow these steps to install it:

1. Insert the current SLSUX02 selection logic in front of label A990 in the SLSUX02D assembler code. The selection logic inserted will receive control for mounts not directed to CA Vtape.
2. Customize and submit HLQ.CCUUJCL(SLSUX02J). The SYSLMOD DD in the LINKEDIT step should point to the CA Vtape SMP/E target loadlib.
3. Concatenate the CA Vtape SMP/E target loadlib in front of the library containing the in-use SLSUX02 module in the HSC started task.
4. Consult the HSC manuals on how to activate the new version of the SLSUX02 exit.

SLSUX08

SLSUX08 influences specific VOLSER mount requests. The CA Vtape sample SLSUX08 exit can be found in HLQ.CCUUSAMP(SLSUX08).

The sample SLSUX08 exit checks the VOLSER prefix to determine if the mount is for a CA Vtape Virtual Volume. When the VOLSER belongs to CA Vtape, the exit sets a return code of 12 to tell HSC the allocation should be handled by a non-silo drive. When a prefix match does not occur, the exit sets a return code of 0 to tell HSC to honor the UNIT information in the JCL.

The sample SLSUX08 exit is the default HSC exit with the following modifications:

```
         USING SLSUX08P,R10          MAP PARM LIST
*        LA    R15,64                INACTIVE EXIT
* ADDED CODE FOR VTAPE PROCESSING
         L     R11,UX08VOLP          LOAD POINTER ADDRESS
         USING SLSUX08V,R11          MAP PARM LIST
         LA    R15,12                USE NON-LIBRARY DRIVES
         CLC   UX08VLSR(1),=CL1'x'   YOUR VIRTUAL TAPE PREFIX NUMBER
         BE    RETURN
         LA    R15,0                 USE LIBRARY DRIVES
         SPACE
RETURN   DS    0H
```

The exit is called after the HSC software has already processed the allocation request and made a decision on what action should be taken. Passing a return code of zero back in the exit tells the HSC software that the decision it has already made is not going to be changed by the exit. If the decision was to intercept the mount, it will be intercepted. If the decision was to not intercept the mount, it will not be intercepted.

If you eject tapes from your silo and do not have any manual or floor drives to mount the ejected tapes on, then you need the mount to occur in the silo and the tape inserted into the silo. A return code of 16 passed back in SLSUX08 informs the HSC software that it should prefer the silo and intercept the mount. This will cause the insert WTOR to be issued.

Consult the *HSC User Exit Guide* for other available SLSUX08 return codes that may be required in your environment.

Once the sample source has been updated for your environment, it must be assembled and linked with the HSC provided JCL and macro libraries and the resulting load module placed in the CA Vtape or HSC loadlib. If the CA Vtape loadlib is used, it must be concatenated in front of the HSC loadlib in the HSC started task. Consult the HSC manuals on how to activate the exit.

Method Three: Update HSC/SMC Parameters and Allocation Routines

We recommend that you implement the POLICY statements as discussed in the section Method One: POLICY Statements (see page 26), to avoid updating your TAPEREQ parameters each time you modify the CA Vtape filters. If you choose to use this method, and your release of the HSC/SMC software still supports these parameters, then you must make the following changes:

- SLSSYS*xx* for the virtual scratch pool tape range.

  ```
  SCRPOOL  NAME(VTAPE) RANGE(V00001-V99999) LABEL(SL)
  ```

  Note: After changing SLSSYS*xx*, you must Recycle the HSC started task to activate the change.

- TAPE*xx* for the data sets to be intercepted by CA Vtape.

  ```
  TAPEREQ  DSN(VTAPE.**) MED(LONG) REC(36) SUBPOOL(VTAPE)
  ```

- VOL*xx* entry for the virtual tape range.

  ```
  VOLATTR SERIAL(V00001-V99999) MEDIA(LONG) RECTECH(36)
  ```

- UNIT*xx* entry for the virtual tape device range.

  ```
  UNITATTR ADDRESS(0F00-0F1F) MODEL(9490)
  ```

Note: Review the HSC manuals to verify the preceding information concerning HSC setup.

If a new esoteric will be defined for CA Vtape, the NNLBDRV HSC parameter in the LIBGEN should be updated with the esoteric as follows:

```
NNLBDRV=VTAPE  or  NNLBDRV=(VTAPE,VTAPE)
```

Where *VTAPE* is the new esoteric on Host0, or Host0 and Host1.

If the parameter already has a value coded, using IBM's HCD software, define a new esoteric containing all the nonlibrary devices and code the NNLBDRV HSC parameter as follows:

```
NNLBDRV=NONSILO  or  NNLBDRV=(NONSILO,NONSILO)
```

Where *NONSILO* is the new esoteric on Host0, or Host0 and Host1.

The new esoteric is only used for the NNLBDRV HSC parameter. The original esoteric and the new CA Vtape esoteric will be used in JCL or software parameters as needed.

Method Four: Use DFSMS or CA Allocate to Perform Unit Replacement

You can avoid the HSC exits and HSC TAPEREQ parameters by using DFSMS and CA Allocate to do unit replacement.

A separate unit esoteric is coded for the CA Vtape Virtual Devices and added to the NNLBDRV HSC parameter as documented in the section Method Three: HSC/SMC Parameters (see page 28). A new data class is defined in SMS and the SMS data class ACS routine or the CA Allocate ASR is updated to assign that data class to data sets that should be intercepted by CA Vtape. The CA Allocate ASR is updated to assign an esoteric containing only the CA Vtape Virtual Devices to the mount request. The new data class is added to the CA Vtape Data Class Filter List.

When a scratch mount is requested, DFSMS or CA Allocate directs it to CA Vtape by assigning the data class and then assigning the CA Vtape esoteric. HSC/SMC ignores the mount because the esoteric does not contain any HSC/SMC controlled devices.

The Virtual Devices still need to be added to the unit esoterics normally used by the applications so that read requests can be properly processed by CA Vtape. Because read requests are for existing data sets, not new data sets, DFSMS, or CA Allocate is not invoked and cannot do unit replacement.

Integration with CA OPS/MVS

CA Vtape provides seamless integration with CA OPS/MVS to provide System State Management (SSM) and health check state management. There is no requirement to define CA Vtape parameters, initialization statements or commands to activate the interface. When both products are active in the same z/OS image, CA Vtape will automatically communicate system and health check status to CA OPS/MVS.

Enable CA OPS/MVS Event Notification

To enable the interface, set the CA OPS/MVS parameter APIACTIVE to ON. This allows CA OPS/MVS to acknowledge and process the events generated by CA Vtape.

System State Management (SSM)

CA Vtape provides a direct interface to the CA OPS/MVS System State Manager (SSM) to notify CA OPS/MVS of the current operating state of the given SVT*n* and SVT*n*V*n* address spaces. This information can then be used to initiate CA Vtape actions as well as actions in other products that are dependent on or have some relationship to CA Vtape.

Note: For more information on using CA OPS/MVS SSM, see the CA OPS/MVS documentation.

Health Check State Management

Each CA Vtape Subsystem issues a heartbeat update every sixty seconds to CA OPS/MVS. This update concerns the current operational health state of the respective Subsystem. If the Subsystem detects a health state change, an immediate update is generated so that CA OPS/MVS has a constant operational health state view of each Subsystem. The current health state information or the lack of a heartbeat update can be used by CA OPS/MVS to initiate actions or warnings concerning a Subsystem or a general system problem.

Note: For more information on using the CA OPS/MVS Health Check application, see the CA OPS/MVS documentation.

Chapter 3: Determining Storage Requirements

This section contains the following topics:

Determining the Global VCAT and BSDS Size

The Global VCAT and BSDS1 are the same size and should be allocated on separate DASD volumes to ensure recoverability. The size of the data sets is based on a fixed area of 7,298 records for control information and one 4KB record for each Virtual VOLSER that needs to be indexed. The following algorithms can be used to determine the size of the Global VCAT and BSDS1:

Control Information Area + VOLSER Area = Total Size

7,298 records + Total VOLSERs = Total Records

608 3390 tracks + (Total VOLSERs / 12) = Total 3390 Tracks

For example, if your Global VCAT will contain 100,000 VOLSERs, the total number of records used in the IDCAMS DEFINE statements and the resulting size of the Global VCAT and BSDS1 would be:

7,298 + 100,000 = 107,298 records

608 3390-tracks + (100,000 / 12) = 8,942 3390-tracks

The JCL to define and initialize the Global VCAT and BSDS will be generated by the Customization Panels later in this chapter. This algorithm is provided to allow you to pick DASD volumes with enough available space to prevent allocation errors.

The Customization Panels will default to indexing 1000 Virtual VOLSERs which is adequate for a trial or demonstration installation. For a live or production installation, a minimum of 100,000 VOLSERs is recommended since the Global VCAT and BSDS cannot be dynamically extended. All Subsystems using a specific Global VCAT and BSDS1 pair must be stopped so these data sets can be reallocated and repopulated.

Local VCAT Size

Each CA Vtape subsystem that is started will require its own Local VCAT. Each Local VCAT is a VSAM linear data set 683 tracks in size.

If you want to start CA Vtape on two systems and you also want to start two instances of CA Vtape on one of those systems, then there will be three active CA Vtape subsystems. You must define three Local VCATs

VDATAQ Data Set Size

Each CA Vtape subsystem that is started dynamically defines a *prefix*.VVE.*sysname*,*subsysnum* data set. *prefix* is the DSN prefix that is defined for CA Vtape use, *sysname* is the system name, and *subsysnum* is the CA Vtape subsystem number (SVT1-8). After defined by the first startup, these data sets are reused by subsequent starts unless they are deleted. Each VDATAQ data set is a VSAM linear data set 690 tracks in size.

Virtual Device Work Data Set Size

Each CA Vtape subsystem that is started dynamically defines a work data set for each Virtual Device that is defined in the OSDRIVES member. These data sets have names in the following format:

prefix.VVE.*Ddevnum*.*sysname*

prefix

Identifies the DSN prefix that is defined for CA Vtape use.

devnum

Identifies the device number.

sysname

Identifies the system name.

After defined by the first startup, subsequent starts reuse these data sets unless the data sets are deleted. Each device work data set is a VSAM linear data set 180 tracks in size.

If you start CA Vtape on two systems and also start two instances of CA Vtape on one of those systems, then there are three active CA Vtape subsystems. If each CA Vtape subsystem has 16 Virtual Devices, then 48 Virtual Device work data sets are defined.

TASKLIB Loadlib

If the Tasklib attribute in the Dynamic Options Section of the OSPARMS Parmlib member is set to Automatic, each CA Vtape subsystem that is started dynamically defines and APF authorizes a run-time loadlib. The STEPLIB DD or link listed loadlib is copied into the run-time loadlib and the Virtual Control Unit started tasks are started using this loadlib.

Determining the Storage Required for Virtual Volume Files

If CA Sales Consulting has already performed a tape study to determine the appropriate amount of network storage required for your Virtual Volumes, you can proceed to the next chapter. If in the future you would like to engage CA Sales Consulting to perform another tape study, contact CA Support or your Sales Representative.

When determining the network storage required for your Virtual Volumes, the first step is to customize and execute the VMAJCL1 member in the HLQ.CCUUJCL. This JCL executes IBM's Volume Mount Analyzer utility.

Understanding the IBM Volume Mount Analyzer (VMA)

If you are unfamiliar with IBM's VMA, read the documentation for the utility and make yourself generally aware of setup and options. The CA Vtape analysis reports only use a small part of the overall capabilities of VMA and we are not trying to reproduce the operational documentation here. For more information, see the IBM publication *DFSMS/MVS Using the Volume Mount Analyzer*.

VMAJCL1 parses your SMF data and uses filters to extract records related to tape mount activity and create an extract file. The creation of an extract file allows multiple analysis reports with different variables to be run without having to reprocess all the SMF log data.

The filters allow you to include or exclude data by jobname, data set name, date, time, unit, and other parameters. Refer to the VMA documentation for specific information about filtering data.

The extract file or files are tersed and sent using FTP to us for processing. Using the VMA filters to eliminate tape mounts for workloads that should not be part of the study will reduce the size of the extract files being transmitted and increase the accuracy of the study results.

Running Product Modeling

You must first determine the location of your SMF log data sets and find the volumes containing records for the period of time your analysis is to be run. You can run the analysis for any period of time and for any number of input LPARs. Include information that reflects all the data you are considering for CA Vtape implementation.

SMF Input Record Types

To produce the desired reports, certain SMF record types are required. If you are not sure what record types are being gathered on your systems, examine the SMF setup specifications or contact the person responsible for setting up the SMF specifications for your installation.

The required SMF record types are 04, 05, 14, 15, 21, 30 (subtypes 4, 5), 34, and 35.

Note: For more information about SMF record types, see your IBM VMA documentation.

How Much Data to Analyze

We recommend that you analyze at least 30 days of SMF data from each of your systems to ensure that an accurate picture of your tape usage, including month end processing, is created. If quarter or year-end processing data is available, running with it will provide an even clearer picture of your tape usage. Running reports for different months or weeks of data will allow you to determine your peak usage times versus overall usage.

JCL Setup

Copy the JCL member VMAJCL1 and make the following changes:

1. Add a job card.
2. Specify the HLQ parameter with a data set name high-level qualifier used to prefix the generated output data set name.
3. Check all marked statements for proper space parameter information. The VMA manual makes specific recommendations regarding these size specifications. Depending on how much data you process, this job can run for an hour or more, so it is important to provide enough space to avoid abends.
4. Specify the data set name and VOLSER information for all input SMF data you want to process.
5. Check the output data set specification to ensure it is valid and is on a volume where it can be saved for later processing.
6. Due to long processing times, confirm the TIME parameter on the JOB and STEP statement to ensure your job is not automatically terminated for lack of time.

7. You can specify VMA statements to limit collection of data to specific time periods, for example, for all the days in one month. Refer to the VMA documentation for details. Input parameters are not required, so the XTRCNTL DD statement may refer to an empty input list.
8. After your changes are complete, submit the job.

Chapter 4: Setting Up SMS

This section describes the SMS setup that is required for Dynamic Cache Management.

This section contains the following topics:

SMS Definitions

The SMS setup requires only the definition of a few constructs and minor changes to the ACS routines.

Define unique DFSMS constructs for use by CA Vtape. Unique constructs shield CA Vtape functionality from that of other products. A change for another product does not accidentally affect CA Vtape.

DFSMS Data Class

CA Vtape requires that a data class with the following characteristics be defined:

```
Volume Count . . . . . .: 2
Data Set Name Type . . . .: EXTENDED
  If Extended  . . . . . .: PREFERRED
  Extended Addressability : NO
  Record Access Bias . . .: USER
Space Constraint Relief  .: YES
  Reduce Space Up To (%) .: 50
  Dynamic Volume Count . .: 2
```

Note: A Record Organization (Recorg) of LS for Linear Data Set is not coded because CA Vtape uses this data class for both VSAM and non-VSAM data set allocations. CA Vtape supplies the appropriate RECORG in its allocation requests.

This list explains each characteristic:

Volume Count

Reserves space in each catalog entry for the number of volumes entered. 2 is recommended to reserve some space in the catalog entry for multiple volume allocations.

Data Set Name Type

Extended ensures that the Data Sets are allocated in extended format. This is required for partial space release.

Extended Addressability

Allows a data set to exceed 4 GBs in size.

Record Access Bias

Allows CA Vtape to control record buffering.

Space Constraint Relief and Reduce Space Up To (%)

These parameters allow DFSMS to modify the space allocation that CA Vtape requests when the storage group becomes constrained for space.

Dynamic Volume Count

Allows a data set to be spanned to this number of DASD volumes if they are available in the storage group.

The other data class characteristics can be left at their default value or blank.

DFSMS Storage Class

CA Vtape requires that you define a storage class with the following characteristics:

```
Performance Objectives
  Direct Millisecond Response . . . :
  Direct Bias . . . . . . . . . . . :
  Sequential Millisecond Response . :
  Sequential Bias . . . . . . . . . :
  Initial Access Response Seconds . :
  Sustained Data Rate (MB/sec)  . . : 0
Availability. . . . . . . . . . . . : N for NOPREF
Accessibility . . . . . . . . . . . : N for NOPREF
  Backup  . . . . . . . . . . . . . :
  Versioning  . . . . . . . . . . . :
Guaranteed Space . . . . . . . . . .: N for NO
Guaranteed Synchronous Write . . . .: N for NO
Multi-Tiered SGs . . . . . . . . . .:
Parallel Access Volume Capability  .: N for NOPREF
Cache Set Name . . . . . . . . . . .:
CF Direct Weight . . . . . . . . . .:
CF Sequential Weight . . . . . . . .:
```

The other storage class characteristics can be left at their default value or blank.

Note: To allow CA Vtape to release unused space, the data sets must not be multistriped. This means that the defined storage class must have a Sustained Data Rate set to zero and Guaranteed Space set to no.

DFSMS Management Class

You do not have to define or assign a management class to CA Vtape data sets. You do not have to back up or migrate the data sets. CA Vtape manages the data set lifecycle.

Alternately, you can define and assign a do-nothing management class to the LDSs. A do nothing management class has the following properties:

- EXPIRE AFTER DAYS NON-USAGE set to NOLIMIT
- EXPIRE AFTER DATE/DAYS set to NOLIMIT
- CMD/AUTO MIGRATE set to NONE
- ADM/USER BACKUP set to NONE
- AUTO BACKUP set to NO

DFSMS Storage Group

A storage administrator can define a unique storage group for CA Vtape data sets.

Because migration and backup of the Dynamic Cache LDSs is not required, set the the Auto Migrate, the Auto Backup, and the Auto Dump fields in the storage group to No.

DFSMS ACS Routines

For Dynamic Cache Management, CA Vtape dynamically allocates a cache data set when needed. The dynamic allocation is requested using the data class and storage class that is defined exclusively for CA Vtape.

If the ACS routines are coded to honor the data class and storage class that a user requests, then no changes are required in the data class or storage class ACS routines. If you set up the ACS routines to override or validate the users' requested data class and storage class automatically, then you modify these ACS routines.

If you defined a new storage group for CA Vtape, then update the storage group ACS to assign the new storage group.

The sample ACS routine changes in this section assume that:

- The CA Vtape DSN prefix that is defined during customization is the default value of CAVTAPE1.
- The data class that is defined for CA Vtape is DCVTAPE.
- The storage class that is defined for CA Vtape is SCVTAPE.
- The storage group that is defined for CA Vtape is SGVTAPE.

The following sample shows a data class ACS change based on the previous assumptions:

```
IF &DSN = CAVTAPE1.VVE.D*.T*.LIB* OR
   &DSN = CAVTAPE1.VVE.VDATAQ.** OR
   &DSN = CAVTAPE1.VVE.D*.* THEN DO
     SET &DATACLAS = 'DCVTAPE'
     EXIT CODE(0)
     END
```

Note: CA Vtape uses the data class (DCVTAPE in the example) to define its DASD data sets. Do not use this data class in the SMS ACS routines to direct tape mounts to CA Vtape. Define a separate data class or classes to intercept tape mounts. If you use a DASD data class to intercept tape mounts, SMS converts the tape mounts to DASD allocations. The allocations then fail, typically with a SPACE error.

The three data set naming patterns being checked for in the sample are as follows:

- The Tasklib loadlib where the qualifier that starts with a D is following by the date, the qualifier that starts with a T is followed by the time, and the last qualifier is followed by the last digit of the Subsystem number (1-8).
- The data set which backs up a data space communication area, saving the stored information across restarts.
- The work data sets allocated for each Virtual Device where the qualifier that starts with a D is followed by the device number and the last qualifier is the system name. These data sets are 8 MBs in size and one is allocated for each online Virtual Device.

Since all the data set names start with CAVTAPE1.VVE, if you do not need to put these data sets in different storage classes or groups, you could code a single pattern of CAVTAPE1.VVE.** in the data class ACS routine.

The following sample shows a storage class ACS change that is based on the previous assumptions:

```
IF &DATACLAS = 'DCVTAPE' THEN DO
     SET &STORCLAS = 'SCVTAPE'
     EXIT CODE(0)
     END
```

The following sample shows a storage group ACS change that is based on the previous assumptions::

```
IF &STORCLAS = 'SCVTAPE' THEN DO
     SET &STORGRP = 'SGVTAPE'
     EXIT CODE(0)
     END
```

The following sample shows a management class ACS change that is based on the previous assumptions::

```
IF &STORCLAS = 'SCVTAPE' THEN DO
     SET &MGMTCLAS = ''
     EXIT CODE(0)
     END
```

Once these changes are activated in DFSMS, test them by allocating a test data set that matches the naming standard of any of these data sets and uses the CA Vtape data class and storage class. After the data set is allocated, execute an IDCAMS LISTCAT to verify that the appropriate DFSMS constructs were assigned to the data set and it was allocated on a DASD volume in the correct storage group.

DASD Allocation Control Products

Only SMS should intercept or modify the CA Vtape dynamic allocations. Because Space Constraint Relief is coded in the DFSMS data class, products that intercept space abends or adjust allocation amounts that are based on DASD pool conditions are not needed. Also, CA Vtape requests specific primary and secondary space values and takes appropriate actions when these amounts cannot be honored. Adjusting these values with another product could result in CA Vtape taking inappropriate actions or an ABEND.

Note: This restriction includes CA Allocate. Update the CA Allocate ASR to ignore all CA Vtape dynamic allocations.

Chapter 5: Using Product Customization Panels

This section contains the following topics:

ISPF Customization Steps

The Customization Process uses input from ISPF panels to create the SUTPARMS configuration member in the PREFIX.SVTJCL data set. The member is then used to define and initialize the control data sets.

The SUTPARMS member contains the variables for the DSN prefix and the Virtual Volume Size. The Virtual Volume Size defaults to 2000 MBs and can be overridden in the OSPOOLS Parmlib member.

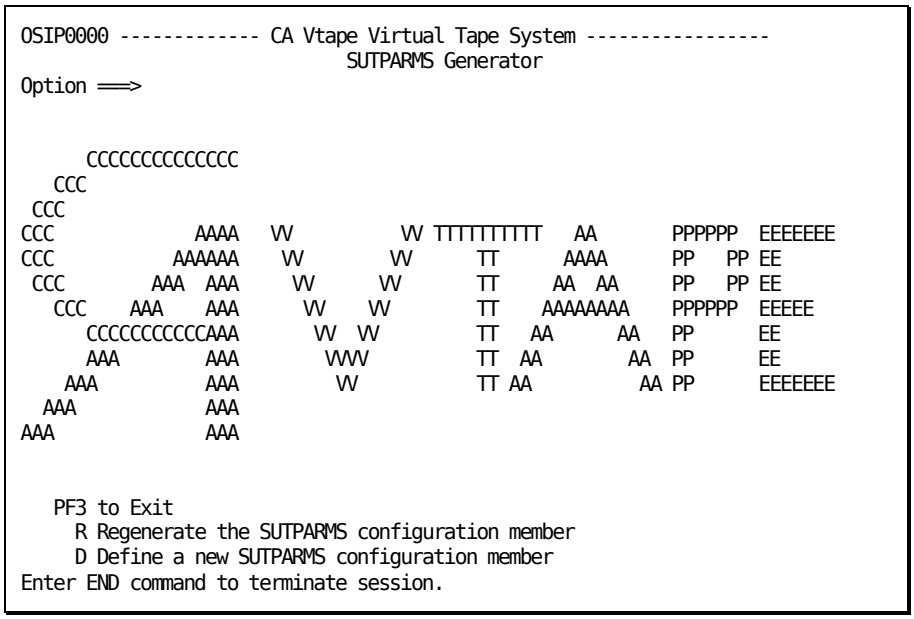
Follow these steps:

1. Type the Following Command in ISPF option 6:

   ```
   EXEC 'HLQ.CCUUEXEC(OSINSTAL)'
   ```

2. Enter **d** to Define a New CA Vtape System on the Option line as follows, and press Enter.

   The following is an example of the main menu.



There are two primary options in the ISPF Customization Panels as shown in the following list:

(R)egenerate

Regenerate the SUTPARMS member for an existing CA Vtape complex. You cannot change the previously entered DSN Prefix. Changing the DSN Prefix would make the SUTPARMS member generated incompatible with an existing CA Vtape Complex. The DSN Prefix can only be changed with the Define option.

Invoke this option to regenerate the SUTPARMS member if it is lost.

(D)efine

Generate the SUTPARMS configuration member used to define and initialize the control data sets in a CA Vtape Complex. Changing the CA Vtape DSN Prefix is permitted.

Invoke this option when defining a new CA Vtape Complex.

Important! Selecting this option and submitting the generated JCL members can cause data loss if access to existing Virtual Volumes is lost. Select this option only to define and initialize a new CA Vtape complex using a new DSN prefix.

3. Enter the PREFIX.SVTJCL library data set name allocated with the VTA08NON job (see page 25). The default SVTJCL data set name is shown. Enter **c** on the command line and press Enter to continue to the next panel.

```
DISP1000  --------------- CA Vtape Virtual Tape System ----------------
                              SUTPARMS Generator


COMMAND ===>  c                                            C to Continue
                                                          PF3 to Previous


  Specify the SVTJCL Data Set Name (Prefix.SVTJCL)

    SVTJCL Library:  CAVTAPE1.SVTJCL_____________________________

  The SVTJCL data set is the repository for the generated
  SUTPARMS configuration member.  The data set was allocated
  by customizing and executing member VTA08NON in the CCUUJCL
  data set.

***
```

4. Enter the CA Vtape DSN Prefix. The default value is CAVTAPE1. Enter g on the command line and press Enter to generate the SUTPARMS configuration member.

   Note: The DSN Prefix is the value that CA Vtape uses to create and catalog work data sets dynamically. This value must match the catalog alias that you defined earlier in the chapter "System Setup."

```
OSIP130D  --------------- CA Vtape Virtual Tape System ----------------
                              SUTPARMS Generator
COMMAND ===>  g                                            G to Generate
Session Mode: Define                                      PF3 to Previous


 The Data Set Name Prefix is used to define CA Vtape
 System Data Sets. A maximum of 23-characters can be entered.


    System DSN Prefix:  CAVTAPE1_______________


    CA Vtape System Data Sets:
      * Prefix.VVE.Ddevn.sysname            Virtual Device Work Data Sets
      * Prefix.VVE.VDATAQ.sysname.subsysnm  Work Save Data Sets
      * Prefix.VVE.Ddate.Ttime.LIBn         Tasklib Loadlib

***
```

   When the generation process is complete, you are placed in a browse session to review the generated SUTPARMS configuration member.

5. Press PF3 to return to the Main Menu. Press PF3 again to leave the Customization Panels or enter a **d** to correct the DSN Prefix.

Chapter 6: Creating CA Vtape Control Data Sets

This chapter provides the information necessary to create the CA Vtape control data sets.

This section contains the following topics:

Define the Global VCAT and BSDS1

Follow these steps:

1. Edit the member GLOBAL found in the SVTJCL library updated from the ISPF customization by EXEC HLQ.CCUUEXEC(OSINSTAL).
2. Add a valid job card and follow the comments in the JCL member to complete its customization.
3. Submit the member to define and initialize the Global VCAT and BSDS1.
4. Review the job output. Verify that all job steps completed successfully. If any of the job steps fail, take corrective actions and rerun that job.

Define the Local VCATs

Follow these steps:

1. Edit the HLQ.CCUUJCL data set member DEFVCAT.
2. Add a valid job card and follow the comments in the JCL member to complete its customization.
3. Submit the member to define and initialize a Local VCAT for one CA Vtape subsystem.
4. If the job ends with a nonzero return code, take the necessary corrective action and resubmit the job.
5. Repeat steps 1 through 4 for each CA Vtape subsystem that will be started. For example, if you will be starting a total of five CA Vtape subsystems, five unique Local VCATs will be required, one for each subsystem.

Chapter 7: Setting Up Tape Mount Intercept Filters

This chapter documents how to set up tape mount intercept filters.

This section contains the following topics:

Data Class and Data Set Name Filters

For CA Vtape to intercept tape mounts, code the Data Class or Data Set Name Filters in the CA Vtape Parmlib filter member. The default Parmlib member name is OSFILTR.

The coded filters are sorted and loaded into the Local VCAT when the CA Vtape Subsystem is started. If you modify the filter member while the Subsystem is active, the program issues the SVT*n* REFRESH=FiLTer for each Subsystem that shares the filter member to trigger a reload.

Tape Mount Redirection Techniques

The topics in this section discuss the advantages and disadvantages of using data class filters, data set name filters, esoterics, or generics for tape mount redirection.

Using DFSMS Data Class Constructs

Use DFSMS Data Class names to redirect a tape mount to a CA Vtape Virtual Device. Using a data class does not mean that the data set is DFSMS-managed.

The advantages of this technique are the opportunity to use the many selection criteria available to DFSMS (DSN, UNIT, JOBNAME, and USERID) and the centralization in DFSMS of all redirection control for both DASD and tape processing.

The disadvantages of this technique are the need for coding changes in the ACS routines, the required security to accomplish such a change, and the increase in complexity of the ACS routines.

For information on using DFSMS and ACS routines, see the IBM *DFSMSdfp Storage Administration* Reference.

Data Set Name Filtering

Data set names or patterns can be used to redirect a tape mount to a CA Vtape Virtual Device.

The advantages of this technique are the ease of implementation (especially for testing purposes) and the centralization of filter processing in a single PDS member. If the filter member is placed in its own data set, access to the data set can be given to application, job scheduling, or operations personnel to update the filters.

The disadvantage of this technique is having only a single selection criterion, data set name.

Both DFSMS Data Class Constructs and Data Set Name Filtering

The CA Vtape Data Class Filter matching takes precedence over the Data Set Name Filter matching. When a data class match occurs, no data set name filter processing is performed. If no data class match occurs, data set name filter processing is performed.

By using both techniques, the advantages of both Data Class and Data Set Name Filtering are combined.

The disadvantages of using both techniques are the additional complexity in the selection logic (especially when the same data set is defined in both filter lists) and the decentralizing of filter processing, which increases your management time. For these reasons, we do not recommend using both techniques.

Esoterics or Generics

CA Vtape intercepts tape mounts by influencing the eligible device table built for the esoteric or generic used in the tape mount request. If the Virtual Devices are not in that eligible device table, CA Vtape will not intercept the mount request even if a filter match occurs. If an intercept was performed in this situation, the mount request would fail with an IEF391I error message or a dynamic allocation error reason code of 04E5 indicating that a tape subsystem eliminated all eligible devices. If your tape devices are over-genned, more devices are defined then actually exist, you may receive a unit required message with a WTOR request that an offline device be brought online.

In this situation the tape mount is ignored to avoid the delays caused by job and task ABENDs. The tape mount is serviced by whatever devices are referenced by the esoteric or generic used. An additional benefit of this protection feature is that it can be used as an additional filtering criterion if application JCL changes are allowed.

For example, a site has 3490 and 3590 tape hardware. The CA Vtape Virtual Devices are defined as 3490 drives and automatically included in the 3490 generic. The Virtual Devices are also added to the esoteric of CART which is the primary esoteric used by the applications. The Virtual Devices are not added to the esoteric of TAPE.

A data set name filter of “PROD./” is added to the CA Vtape filters. This filter would cause all scratch mounts using UNIT=3490 or UNIT=CART for data sets with a high-level qualifier of PROD to be intercepted. However, if UNIT=3590 or UNIT=TAPE was used for data set PROD.MASTER.FILE, it would not intercept it because the Virtual Devices are not in the 3590 generic or the TAPE esoteric.

After deciding which applications will use Virtual Volumes, the JCL and tasks for that application can be reviewed to ensure they use UNIT=3490 or UNIT=CART and the single data set name filter will cause all of their tape mounts to be intercepted and directed to Virtual Volumes. If these applications have specific tape data sets that fill 3590 cartridges, they can continue to use UNIT=3590 or UNIT=TAPE and not be intercepted.

Some sites have chosen to control the use of CA Vtape by using JCL changes. They create an esoteric of VTAPE containing only the Virtual Devices and code a single data set name filter of “/”. As applications and tape data sets were reviewed and a decision was made to send them to Virtual Volumes, the task or JCL was changed to use the VTAPE esoteric.

This technique greatly simplifies filter coding at the cost of requiring JCL changes.

Tape Mount Intercept Filters

There are two types of filters: includes and excludes. Include filters tell CA Vtape which tape mounts to intercept. Exclude filters work with include filters to tell CA Vtape which tape mounts not to intercept.

Include filters can be coded for data set names, data set name patterns, and SMS data classes. Exclude filters can only be coded for data set names and data set name patterns. Pattern masking is not allowed for data class filters so data class exclude filters are not needed or allowed.

OSFILTR Parmlib Member

The Parmlib Directory Section in the OSPARMS Parmlib member contains attributes which identify which Parmlib members contain specific sets of related Parmlib attributes. One of the attributes is DatasetFilters which has a default value of OSFILTR. The OSFILTR Parmlib member is where you code your tape mount intercept filters.

OSFILTR contains a number of comment lines documenting the structure and syntax of the member, how to customize the member, and examples. The member contains an index section and one or more filter sections which contain include or exclude filters.

The index section name is <DatasetFilters>. In this section, you code IncludeDatasets, ExcludeDatasets, and IncludeDataClass attributes. Code these attributes to organize and document your filter sections properly. The values of these attributes are unique, customized section names which contain filters of the type that is associated with the attribute name.

For example, IncludeDatasets = Production_Include_Data_Sets, would indicate that a section named <Production_Include_Data_Sets> is coded in the member and it contains include filters. CA Vtape uses the values coded in this section to try and intercept tape mounts. Whether the entries are really for production data sets will not be validated. The customizable nature of the names allows you to document what the sections are for.

These filter sections contain GROUP*nn* attributes where *nn* is 01-04, 11-14, 21-24, and so on, up to 71-74. The value of the group attribute is a fully qualified data set name, a data set name pattern, or a data class.

Groups are documented in the OSGROUP Parmlib member. Groups are used to assign management policies to tape data sets that have similar requirements. All production tape data sets could be assigned to group 01. All test tape data sets could be assigned to group 02. All disaster recovery tape data sets could be assigned to group 03 allowing you to manage these data groups differently.

The best practice when starting out is to assign all tape data sets to a single group. When you are more familiar with CA Vtape and the effects of splitting the tape data into multiple groups, the filters and groups can be modified.

The following is an example of a typical filter setup:

```
<DatasetFilters>
  IncludeDatasets = Prod_Data_Set_Includes
  ExcludeDatasets = Prod_Data_Set_Excludes
  IncludeDatasets = Test_Data_Set_Includes
<Prod_Data_Set_Includes>
      Group01 = PROD./
<Prod_Data_Set_Excludes>
      Group01 = PROD.PAYROLL./
<Test_Data_Set_Includes>
      Group02 = TEST./
```

Use TSO edit to add, delete, and modify filter entries. Use the SVT*n* REFRESH=FiLTers console command to notify the appropriate CA Vtape Subsystem of filter changes.

Note: The SVT*n* REFRESH=FiLTers console command is local in scope. If three Subsystems running on two Logical Partitions use the same filter member that was changed, issue the console command once for each Subsystem.

Date Set Name Pattern Masking

The following wildcard characters can be used in Data Set Name Filter entries:

?

If this character is in the data set name string, it matches any nonblank character within a character string, for example, TEST?.DATA matches TESTA.DATA and TEST1.DATA. TEST.DATA would not match this pattern.

*****

If this character is present in a data set name string, a single-level node is not checked, for example, TEST.*.DATA or *.TEST.*.DATA. TEST.DATA would not match these patterns.

The asterisk can be placed after significant characters in a node to indicate that any following characters in the node are acceptable, for example, TEST.A*.DATA.

Note: In CA Vtape an asterisk or double asterisks are not coded to indicate any data set name or any number of characters beyond this point in the string. The forward slash (/) is used instead.

/

When this character is in a data set name pattern, comparison to the input string terminates at the previous character. These are named prefix entries. If the prefix matches the input string up to the slash, the comparison is satisfied. For example, SYS/ matches all data sets whose names begin with SYS, regardless of what follows. This usage is similar to the way that IBM products use the * character. In IBM products, the DEPT751* pattern finds all data sets that start with DEPT751. To obtain the same result in CA Vtape, use the DEPT751/ pattern.

!

When the program encounters this character (English exclamation mark, X"5A"), the program searches the input for a match on the characters that follow. The pattern characters can occur anywhere in the input string. For example, the pattern !SYS1 matches any data set name that contains the SYS1 string anywhere in the name. TEST.SYS1, SYS1.TEST, and TEST.DFSYS12.DATA would all match this pattern.

Note: Wildcarding is not allowed for Data Class Filters.

When CA Vtape is started, the program issues the following messages to document the type and number of filters coded:

```
SVT1R3500I Enhanced <IncludeDataClass> Filters . . . . .     1
SVT1R3500I Enhanced <IncludeDataSets> Filters. . . . . .    54
SVT1R3500I Enhanced <ExcludeDataSets> Filters. . . . . .     9
```

If the program does not encounter any include filter entries, the program issues a warning message. Without include filters, CA Vtape cannot intercept any scratch tape mounts.

```
SVT1R3502I WARNING: Enhanced Filter List has no include entries.
```

Filter Processing

Filters are processed as follows:

1. Data class includes are processed first. There is no interaction between data class includes and data set name includes or excludes.
2. If a data class match is not found, then data set name includes are processed.
3. If a data set name include match is found, the data set name excludes are processed.
4. If an exclude for the same group as the include is found, the data set name include scan will be redriven to determine if any subsequent includes would match for another group.
5. If another include match occurs, the exclude scan will be redriven. These actions will repeat until all the includes have been scanned without a match or all excludes have been scanned without a match.

Consider the following sample Parmlib filter member:

```
<DatasetFilters>
IncludeDataSets  = IncludeDSNames
ExcludeDataSets  = ExcludeDSNames
IncludeDataClass = IncludeDataClass

<IncludeDSNames>
GROUP01='SYS?./'
GROUP02='SYS2./'

<ExcludeDSNames>
GROUP01='SYS2./'

<IncludeDataClass>
GROUP11=VTAPEG11
```

A tape data set assigned the VTAPEG11 data class by DFSMS will be intercepted and assigned to group 11. No other filter processing is done.

A tape data set that is not assigned the VTAPEG11 data class and has a name starting with “SYS” will initially match the GROUP01 “SYS?./” include entry. If the data set name does not start with SYS2, no exclude entry will be matched and the data set will be intercepted and assigned to group 1.

If the data set name starts with SYS2, the match to group 1 is rejected by the GROUP01 “SYS2./” exclude entry. The include scan will resume and match the GROUP02 “SYS2./” include entry, and the data set will be intercepted and assigned to group 2.

Any other data set name will not match an include entry and will not be intercepted.

The filter entries are sorted by name or pattern before being loaded into the Local VCAT. Consequently, you cannot rely on the order of entry in the Parmlib filter member to determine which filter entry will be used to intercept the data set.

Consider the following sample filter member:

```
<IncludeDSNames>
 GROUP01='SYS2./'
 GROUP02='SYS?./'
```

This member will not cause all SYS2 data sets to be assigned to group 1 while all other SYS data sets are assigned to group 2. When sorted, the GROUP02 pattern will precede the GROUP01 pattern. As a result, all data sets with names starting with SYS, including SYS2 data sets, will be assigned to group 2. An exclude filter for GROUP02 must be coded to prevent the SYS2 data sets from being intercepted by that group as in the following example:

```
<IncludeDSNames>
 GROUP01='SYS2./'
 GROUP02='SYS?./'

<ExcludeDSNames>
 GROUP02='SYS2./'
```

Chapter 8: Synchronizing Scratch Tapes

This chapter describes the synchronization of scratch tapes between the installed Tape Management System and CA Vtape.

This section contains the following topics:

Tape Expiration

The installed tape management system controls the expiration of the Virtual Volumes. The Virtual Volumes are under the tape retention control that the application requested in their JCL or dynamic allocation request. The Virtual Volumes can also be under the retention that the tape management system specified in the retention rules.

When Virtual Volumes meet the required retention control requirements, the tape management system expires and scratches the affected tapes. Virtual Volume expirations must be communicated to CA Vtape so that its control data sets can be updated with the change in status.

Scratch Tape Synchronization

As Virtual Volumes are expired, CA Vtape must be notified for the following actions to be taken:

- Changing the status of the Virtual Volumes in the Global VCAT.
- Renaming of the Virtual Volume files on the Linux Server. The program deletes the renamed files after three days, by default.

In general, notification is performed by executing a batch Scratch Synchronization Job specific to the installed Tape Management System. If other scratch synchronization processes are available, they are discussed in the section specific to the installed Tape Management System.

The Scratch Synchronization Job should be scheduled to run after the normal Tape Management System expiration cycle has completed.

Because the Global VCAT is shared, run the Scratch Synchronization Job only once in a CA Vtape Complex.

Important! When multiple CA Vtape Complexes are installed, execute all appropriate Scratch Synchronization Job steps. Execute the steps in the order provided in the job and only once for each Complex. Otherwise, you can deplete available scratch tapes or scratch active Virtual Volumes, causing a data loss.

The following sections document the scratch synchronization process and other requirements for specific Tape Management Systems. Review the section for your Tape Management System and take the required actions.

CA 1 Tape Management System

Two methods are available to notify CA Vtape when a Virtual Volume has expired.

- The CA 1 ROBSCR option (see page 59).
- The Scratch Synchronization Job (see page 59).

CA 1 ROBSCR Option

The CA 1 TMSCLEAN job returns volumes to scratch status. If the ROBTY fields of the Virtual Volume VOLSER range in the CA 1 TMC have been updated to a value of x'88 and the ROBSCR option is set to Y, TMSCLEAN will automatically notify CA Vtape as each Virtual Volume is expired.

For this notification to occur, the TMSCLEAN program must have access to the CA Vtape CCUULOAD data set by adding a STEPLIB DD to the TMSLCEAN JCL or by adding the CCUULOAD data set to the link list.

Note: The Scratch Synchronization Job should still be customized and executed once a week or month to ensure that all scratched Virtual Volumes are kept in-sync.

CA 1 Scratch Synchronization Job

The Scratch Synchronization Job generates a CA 1 scratch report to determine which Virtual Volumes are in scratch status. Then the job updates those Virtual Volume entries in the Global VCAT that require a status change.

Sample JCL for the CA 1 scratch synchronization job can be found in HLQ.CCUUJCL(CA1). Follow the JCL comments to customize the job with the appropriate CA 1 options data set. You can also customize with the prefix character or characters of the defined Virtual Volume VOLSER ranges. If you use the CA 1 ROBSCR option, you still customize the Scratch Synchronization Job and execute it once a week or month to ensure that all scratched Virtual Volumes are kept in-sync.

TMSCLEAN only notifies CA Vtape when a Virtual Volume is scratched. If that notification fails, TMSCLEAN does not issue another notification. The Scratch Synchronization Job requests a CA 1 scratch report so it always checks all Virtual Volumes in scratch status to ensure that they are in-sync.

CA TLMS Tape Management

The scratch synchronization job requests a CA TLMS scratch report to determine which Virtual Volumes are in scratch status and updates the Virtual Volume Entries in the Global VCAT that require a status change.

Sample JCL for the CA TLMS scratch synchronization job can be found in HLQ.CCUUJCL(TLMS). Follow the JCL comments to customize the job and update the HLQ.CCUUEXEC(TLMSCE) program with the defined Virtual Volume VOLSER ranges.

DFSMSrmm

The scratch synchronization job executes the RMM SEARCHVOLUME command to determine which Virtual Volumes are in scratch status and updates those Virtual Volume Entries in the Global VCAT that require a status change.

Sample JCL for the DFSMSrmm scratch synchronization job can be found in HLQ.CCUUJCL(RMM). Follow the JCL comments to customize the job with the defined Virtual Volume VOLSER ranges.

Control-T (BMC)

The scratch synchronization job generates a CTT scratch report to determine which Virtual Volumes are in scratch status and updates those Virtual Volume Entries in the Global VCAT that require a status change.

Sample JCL for the Control-T scratch synchronization job can be found in HLQ.CCUUJCL(CTT). Follow the JCL comments to customize the job with the appropriate report parameters and the defined Virtual Volume VOLSER ranges.

AutoMedia (Zara)

The scratch synchronization job generates an AutoMedia scratch report to determine which Virtual Volumes are in scratch status and updates those Virtual Volume Entries in the Global VCAT that require a status change.

Sample JCL for the AutoMedia scratch synchronization job can be found in HLQ.CCUUJCL(ZARA). Follow the JCL comments to customize the job with the appropriate report parameters and the defined Virtual Volume VOLSER ranges.

Other Tape Management Systems

If the installed tape management system is not one previously listed in this chapter, review the section CA 1 Tape Management System. Use the sample JCL provided for CA 1 to build a scratch synchronization job.

The only specific requirement for the job is that the file of VOLSERs in scratch status must have one VOLSER per record with the VOLSER beginning in column one. Other information can appear on the rest of the line for reference purposes and will be ignored by the scratch synchronization process.

The maximum supported LRECL (logical record length) for the scratch VOLSER file is 255 bytes. Record format can be F or FB.

If your tape management system cannot produce a report listing the VOLSERs in column one, contact CA Support for help.

Chapter 9: Processing Foreign Tapes

This chapter describes foreign tape processing.

This section contains the following topics:

Foreign Tape Processing

A foreign tape is a tape that is not defined to the Tape Management System. Foreign tapes fall into two categories:

- VOLSERs that were never defined to the Tape Management System
- duplicate VOLSERs.

Foreign tape processing involves coding a special value in your JCL. The value indicates that the Tape Management System does not track and validate the use of the VOLSER referenced. The special value is typically LABEL=EXPDT=98000.

CA Vtape has similar capability that the ForeignTapesExpdt attribute (in the Dynamic Options Section of Parmlib) controls. When a specific mount request for a CA Vtape VOLSER is detected and the referenced DD statement defines an expiration date that matches the ForeignTapesExpdt value, CA Vtape ignores the mount request.

If you write to a VOLSER that matches that of a CA Vtape Virtual Volume, all subsequent mount requests for this VOLSER must use a LABEL=EXPDT value equal to the ForeignTapesExpdt value. Otherwise, CA Vtape tries to intercept the mount request and mount the Virtual Volume instead of the original VOLSER written.

Define a ForeignTapesExpdt value consistent with that used by the installed tape management system. For example, when mounting a tape that CA 1 does not control, code LABEL=EXPDT=98000 on the DD statement to bypass CA 1 validation. Using the same value for CA Vtape allows a foreign tape with a VOLSER matching that of a Virtual Volume to be mounted on a non-Virtual Device.

To correct problems, you might need to mount a Virtual Volume but bypass tape management validation. Set the ForeignTapesExpdt attribute to a value that does not match that of the tape management system. To pick up the change, issue the SVT*n* REFRESH=OPTIONS console command. Code LABEL=EXPDT=98000 in your JCL to bypass the tape management system, but not CA Vtape.

Another way to mount a foreign tape whose VOLSER matches that of a CA Vtape Virtual Volume is to code a unit esoteric or generic that does not contain the Virtual Devices or code a device number that does not belong to CA Vtape. CA Vtape can only intercept a tape mount when its Virtual Devices are part of the eligible device list associated with the unit esoteric or generic being used or a Virtual Device number is specifically referenced. All other mounts are ignored.

Chapter 10: Configuring the Parameter Library (Parmlib)

This chapter provides the basic structure and syntax of the parameter library members, how to modify the members, how to verify the syntax of your changes, and the initial customization that should occur after the product is installed.

This section contains the following topics:

Parmlib Description

The Parmlib is a partitioned data set with fixed or fixed-block 80-byte logical records: DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400,DSORG=PO). The Parmlib contains one or more members that contain CA Vtape parameters that are known as attributes. The related attributes are grouped into sections within a Parmlib member.

Sample Parmlib members containing the recommended settings for the CA Vtape attributes can be found in the HLQ.CCUUPARM library. The members are as follows:

OSPARMS

Parmlib member directory, startup options, and dynamic options.

OSDRIVE

Virtual tape control unit and drive definitions. Optional CHPID device list

OSGROUP

Group definitions and policies that are applied to the Virtual Volumes.

OSFILTR

Tape mount intercept filter lists.

OSPOOLS

Volume pools and VOLSER ranges. Optional Virtual Volume Size.

OSSCMDS

Startup commands.

OSPCMDS

Shutdown commands.

These members together comprise one complete set of all the attributes that CA Vtape uses. Each member contains one or more sections containing information that CA Vtape uses at startup, shutdown, or during normal operations. This information is split into multiple members to make management and maintenance easier. If desired, you can combine one or more members or all members into a single member.

Parmlib Syntax

Each of the sample Parmlib members contains comments that document the member structure, how to customize that member, the member attributes, and valid attribute values.

In general, Parmlib members consist of the following:

Blank lines

Use blank lines for legibility. The program ignores blank lines.

;Comments

Comments always begin with a semicolon. Comments can start in any column. These statements are used for documentation, and readability. The program ignores comments.

<SectionNames>

Section names are always enclosed with greater than and less than signs ('<' and '>'). Sections are used to identify a common set of related Parmlib attributes. The order of sections within a Parmlib member is not important. Sections cannot be embedded or nested. Each new <SectionName> marks the end of the prior section.

Attribute=Values

The left side of the equal sign denotes the name of a particular attribute while the right side denotes its assigned value or values.

Observe the following:

- Entries are not case-sensitive.
- No line continuation character exists.
- Attribute=value pairs must be coded together on a single line between columns 1 and 72.

- Descriptive attribute names are used to associate and assign data values to the Parmlib parameters.
- Values containing blanks or other special characters must be enclosed within single (') or double (“) quotation marks. For example, in member OSGROUP, to define the value. Production Files to the Description attribute, you will code ‘Production Files'.
- Attributes must be coded in their assigned section.
- You can concatenate multiple Parmlib data sets. The program uses the first member in the concatenated data sets that contains the searched section name to load the corresponding attribute values.

Parmlib and Symbolic Substitution

CA Vtape supports symbolic substitution in Parmlib member names and Parmlib attribute values. This allows you to run multiple CA Vtape Subsystems on a single LPAR or multiple LPARs using a single Parmlib.

Parmlib Attribute Values

CA Vtape supports system symbol substitution within the Parmlib members. For example, &SYSNAME resolves to the system name on which a Subsystem is executed. In addition to the standard system symbols, the following CA Vtape symbols may be used:

&JOBNAME

For the jobname of the CA Vtape started task.

&NULL

Nulls, a zero length string.

&PARMDIR

For the Parmlib member containing the Parmlib Directory section.

&STEPNAME

For the stepname of the CA Vtape started task.

&SVTS

For the subsystem id of the CA Vtape started task (SVT*n* where *n* = 1-8).

&THIS

For the current Parmlib member name.

Symbolic variables allow you to assign unique values to attributes while sharing parmlib members among different Subsystems.

For example, you define IBM logstreams on LPARS named SYS1 and SYS2 to offload and save the internal logger data. If you defined the logstream names as VTAPE.SYS1.LOGSTREAM and VTAPE.SYS2.LOGSTREAM, you do not need to have two Dynamic Options Sections members so you can set two different values for the Logstream attribute.

All that is necessary is to edit the OSPARMS member of the parmlib, find the Logstream attribute and set it to a value of VTAPE.&SYSNAME.LOGSTREAM:

Logstream = VTAPE.&SYSNAME.LOGSTREAM

When you start CA Vtape, the program substitutes the system name for the &SYSNAME variable and the program accesses the appropriate IBM logstream for that LPAR.

The MessagePrefix attribute is an example of a parmlib attribute that defaults to a symbolic substitution value.

When yous start CA Vtape, it dynamically defines itself as a subsystem to the operating system. The subsystem id used is taken from the SVTS parameter on the EXEC statement of the SVTSCE PROC. When started with the SVTS parameter set to SVT1, the MessagePrefix value becomes SVT1. This Subsystem will preface all of its messages with SVT1. If the SVTS parameter is set to SVT2, the MessagePrefix value becomes SVT2. This Subsystem will preface all of its messages with SVT2.

Symbolic Substitution for Parmlib Member Names

You can start CA Vtape on two LPARs, SYS1 and SYS2, with different settings for attributes in the Dynamic Options Sections. Make the following changes with a single Parmlib:

1. Edit the OSPARMS member and copy the Dynamic Options Section out to two new members in Parmlib that is named VTDOSYS1 and VTDOSYS2.
2. Find the Parmlib Directory Section in the OSPARMS member and change the DynamicOptions attribute from &THIS to VTDO&SYSNAME.

   DynamicOptions = VTDO&SYSNAME
3. Edit the new VTDOSYS1 and VTDOSYS2 Parmlib members and set the LPAR-specific values for the Dynamic Options Section attributes.

When started on SYS1, the DynamicOptions value of VTDO&SYSNAME resolves to VTDOSYS1 and the initialization reads that member to load the Dynamic Options Section values. When started on SYS2, the DynamicOptions value of VTDO&SYSNAME resolves to VTDOSYS2 and the initialization reads that member to load the Dynamic Options Section values.

Symbolic substitution can be tailored to select only part of a substitution variable. For example, you are starting an SVT1 and an SVT2 on LPARs SYSA and SYSB. You could change the Parmlib Directory Section DynamicOptions attribute to DO&SYSNAME.&SVTS(4:1):

DynamicOptions = DO&SYSNAME.&SVTS(4:1)

&SVTS(4:1) indicates that starting with the fourth character of the SVTS parameter, a one-character substitution should occur. If the SVTS parameter value is SVT1, 1 would be substituted. If the SVTS parameter value is SVT2, 2 would be substituted.

For DO&SYSNAME.&SVTS(4:1), when SVT1 is started on LPAR SYSA, the DynamicOptions attribute resolves to a value of DOSYSA1. When SVT2 is started on LPAR SYSA, the DynamicOptions attribute resolves to a value of DOSYSA2.

For complete information regarding the use of system symbols including substring notation, see the IBM MVS Initialization and Tuning Reference.

Initial Configuration of the Parmlib

Customize the Parmlib that was allocated and populated by running the VTA08NON job in the chapter "System Setup".

For more information about the Parmlib attributes and their values, see the chapter "The Parameter Library (PARMLIB)." The sample Parmlib member attribute settings are typically acceptable at any site, but review them to ensure they will work at your site. Where applicable we direct you to a specific attribute to make a recommended change.

Follow these steps:

1. Edit OSPARMS in PREFIX.PARMLIB.

   Follow the comments in the member to set up the attributes in the Startup Options, Dynamic Options, and Open System Server sections.

   Review and update the following Parmlib attributes:

   Startup Options Section

   - CacheDefaultDataClass to the data class name you defined.
   - CacheDefaultStorageClass to the storage class name you defined.
   - DsnameBSDS1 to the name of the BSDS1 data set you defined.
   - DsnameGlobalVcat to the name of the Global VCAT you defined.
   - DsnameLocalVcat to the name of the Local VCAT you defined. Using symbolic substitution is recommended. If you followed the recommended naming standard, a value of ‘CAVTAPE1.&SYSNAME..&SVTS.VCAT’ would be used.
   - MessagePrefix leave at the default value of &SVTS.

- SubAddressSpacename leave at the default value of &JOBNAME.
- Tasklib to Automatic which dynamically allocates and APF authorize a loadlib data set for use when starting and restarting the Virtual Device Controllers address spaces. If you are executing IBM health checks, this can provoke messages. See Library Mode for this attribute in the chapter "The Parameter Library (PARMLIB)" to prevent the health check messages.
- ZIIPExploitation to Y if you have a zIIP engine that is installed or want RMF to track potential zIIP usage.

Dynamic Options Section

- ForeignTapesExpdt to the same value that your tape management system uses.
- HardwareCompressionOption to N if the network storage you use already performs compression or data deduplication.
- Logstream with the IBM system logger log stream name defined for CA Vtape use.
- PercentRunOnzIIP to 100 if you are tracking potential zIIP usage or to an appropriate percentage if you have a zIIP engine installed. Discussing this setting with your z/OS or DB2 Performance and Tuning Specialist is recommended.
- ScratchVolumesThreshold. Review and set the value high (for example, 5000). The high value ensures that you are warned of a pending shortage early. The high value also provides the time to determine the number of Virtual Volumes you use and scratch daily so you can calculate a better threshold value.

Open System Server Section

- The IPAddress and Port attributes assigned appropriate values that are used for communication through TCPIP with the CA Cloud Linux Server. This allows the Subsystem to query status and asynchronously communicate requests with the CA Cloud Linux Server.
- The PipeUnitAddress attribute assigned the CTC device addresses used to communicate Virtual Volume data with the CA Cloud Storage for System z Server. These CTC connections provide secure and high performance transmission of data with the CA Cloud Linux Server.

2. Edit OSPOOLS.

   Follow the comments in the member to customize the Volume Pool Definitions and Volume Pool Sections.

   Note: If you have a tape management system, define the VOLSER ranges in the tape management system to let the system track and control the use of the Virtual Volumes like physical tapes. Reserve the ranges that you define solely for the use of CA Vtape.

   Define the subset of VOLSERs reserved for disaster recovery testing in Pool8.

3. Edit OSDRIVE.

   Follow the comments in the member to:

   - Split the primary set of Virtual Devices between controller 1 and controller 2. If needed, up to eight controllers can be defined.
   - Uncomment and update the ChpidDeviceList attributes with the list of devices with CHPIDs that Virtual Devices use. Use the device addresses of OSA cards or for DASD volumes in a different DASD subsystem.

     Note: For more information about choosing CHPIDs, see the section Virtual Device Channel Paths in the chapter "Virtual Device Engine".

4. Create OSDRALT.

   Copy OSDRIVE into OSDRALT and change the primary set of Virtual Devices to the alternate set. Use this member to avoid an IPL if the operating system boxes the current set of Virtual Devices and will not let them be varied back online after they have been restarted in CA Vtape.

5. Edit OSGROUP.

   Follow the comments in the member to set up the group options.

   Note: Pool8 is reserved for disaster recovery testing. Do not code Pool8 on the VolumePools attribute for any active group.

6. Edit OSFILTR.

   Follow the comments in the member to add one or more Data Set Name Filters or Data Class Filters.

   If you enter Data Class filters, make the corresponding changes in the DFSMS ACS routines. This assigns the entered Data Classes to data sets whose tape mounts CA Vtape intercepted.

   Note: Do not add the CacheDefaultDataClass value as one of the data class filters.

7. Secure the SUTPARMS member.

   Copy the generated SUTPARMS member from the SVTJCL library to the Parmlib. This ensures that the member, which contains the internal configuration data that CA Vtape needs to initialize new control data sets, is not accidentally overlaid or deleted and is available at your disaster recovery site.

Parmlib Syntax Verification

At startup, CA Vtape reads and validates the Parmlib members that define run-time attributes. Parmlib syntax errors can prevent CA Vtape from starting up properly and can result in one or more error messages describing the condition and a subsequent ABEND.

Because Parmlib changes can be critical to the performance and operation of CA Vtape, it is important to review all such changes. To help you make and test your changes, a Parmlib Syntax Check Utility has been provided. The utility invokes the CA Vtape Parmlib reader to verify the syntax of the Parmlib members. The utility produces a report that details the attributes read and their values.

Customize and submit the HLQ.CCUUJCL(SVTPARMS) member. The job output contains a report of each Parmlib member successfully read and stops when a syntax or structure error is encountered. The resulting error message indicates the Parmlib member, section, attribute, line and column number of the error, and the validation program maintenance level.

The following is a sample Parmlib validation error message:

```
SVTSQQ136E Parmlib error:  777
Invalid option for this Attribute
  CacheManagement = DYNAMC
                   >*<<-- See Line.Col(74.21)
Program SVTSPRMS CCUUC60  ~RMID(RO54767)
PSW@(X'205CEB48') Offset@(X'009E80') Return.Reason(12.4)
DD(SVTPARMS) Member(OSPARMS) Section<StartupOptions>
```

Correct any errors and resubmit the utility until you receive a zero return code.

Automatic Command Execution

CA Vtape lets you automatically execute console commands during startup and shutdown: operating system, JES2, CA Vtape, and other console commands.

To use this feature, add your console commands to member names coded for the StartupCommands and ShutdownCommands attributes in the Parmlib Directory Section in the Parmlib OSPARMS member.

The sample members are OSSCMDS and OSPCMDS for startup and shutdown respectively.

Chapter 11: The Parameter Library (Parmlib)

This section contains the following topics:

Parmlib Sample Members, Format, Syntax, Attributes, and Tables

This chapter documents the Parmlib sample members, their format, their syntax, the customizable attributes they contain, and tables which document what Parmlib member the attributes are located in and what CA Vtape function each attribute is associated with.

Parmlib Attribute Feature and Location

The following table of Parmlib attributes documents where each Parmlib attribute is located and the product features it is related to. Multiple product features can use the same attributess. Some attributes are in multiple Parmlib members.

Consider the following:

- An attribute listed with a format of 'xyz | x' indicates the full attribute name followed by its abbreviation.
- The sample Parmlib members can be found in HLQ.CCUUPARM.
- The Location value documents the sample Parmlib member and the section in that member which contains the attribute.

Attribute	Feature	Location
AllowBTEencryptionForVVE	VirtualDeviceEngine	OSPARMS DynamicOptions
AllowConcurrentVVEReadAccess	VirtualDeviceEngine	OSPARMS StartupOptions
BypassUCBChecking	VirtualDeviceEngine	OSPARMS StartupOptions
CacheDefaultDataClass	Miscellaneous	OSPARMS StartupOptions
CacheDefaultStorageClass	Miscellaneous	OSPARMS StartupOptions
CachePrimary	Miscellaneous	OSPARMS DynamicOptions

Attribute	Feature	Location
CacheSecondary	Miscellaneous	OSPARMS DynamicOptions
ChpidDeviceList	VirtualDeviceEngine	OSDRIVE Virtual Device List
Command | Cmd	Miscellaneous	■ OSPCMDS ShutdownCommands ■ OSSCMDS StartupCommands
ConsoleCommandTimeout	Miscellaneous	OSPARMS DynamicOptions
DatasetFilters	ParmlibStructure	OSPARMS ParmlibDirectory
DefaultGroup	Miscellaneous	OSPARMS DynamicOptions
Description	Miscellaneous	OSGROUP GroupSections
DsnameBSDS1	Miscellaneous	OSPARMS StartupOptions
DsnameGlobalVCAT	Miscellaneous	OSPARMS StartupOptions
DsnameLocalVCAT	Miscellaneous	OSPARMS StartupOptions
DsnameSYSTCPD	Network DASD Storage	OSPARMS OpenSystemServer
DynamicOptions	ParmlibStructure	OSPARMS ParmlibDirectory
ExcludeDataSets	Filtering	OSFILTR DataSetFilters
ForeignTapesExpdt	Miscellaneous	OSPARMS DynamicOptions
GlobalReserve	Miscellaneous	OSPARMS DynamicOptions
GroupDefinitions	ParmlibStructure	OSPARMS ParmlibDirectory
Group01-74	■ Filtering ■ Group Policies	■ OSFILTR FilterSections ■ OSGROUP GroupDefinitions
HardwareCompressionMethod	DASD Buffer Compression	OSPARMS DynamicOptions
HardwareCompressionOption	DASDBufferCompression	■ OSPARMS DynamicOptions ■ OSGROUP GroupSections
IncludeDataClass	Filtering	OSFILTR DataSetFilters
IncludeDataSets	Filtering	OSFILTR DataSetFilters
IOCpuTimeout	VirtualDeviceEngine	OSPARMS DynamicOptions
IPAddress	Network DASD Storage	OSPARMS OpenSystemServer
LogCSASize	Logger	OSPARMS StartupOptions
LogDataspaceSize	Logger	OSPARMS StartupOptions
LogDetailLevel	Logger	OSPARMS DynamicOptions
LogStream	Logger	OSPARMS DynamicOptions

Attribute		Feature	Location
	MaximumCompressionCPU	Virtual Volume Compression	■ OSPARMS DynamicOptions ■ OSGROUP GroupSections
MessagePrefix		Miscellaneous	OSPARMS StartupOptions
MIHTimeoutValue		Virtual Device Engine	OSPARMS StartupOptions
	MinimumCompressionRate	Virtual Volume Compression	■ OSPARMS DynamicOptions ■ OSGROUP GroupSections
Mode		Miscellaneous	OSPARMS Architecture
MountRejectThreshold		VirtualDeviceEngine	OSPARMS DynamicOptions
Offline		VirtualDeviceEngine	OSDRIVE VirtualDeviceList
Online		VirtualDeviceEngine	OSDRIVE VirtualDeviceList
PercentRunOnZIIP		VirtualDeviceEngine	OSPARMS DynamicOptions
PipeUnitAddress		Network DASD Storage	OSPARMS OpenSystemServer
Pool1-8		VirtualDeviceEngine	OSPOOLS PoolSections
Port		Network DASD Storage	OSPARMS OpenSystemServer
ProtocolFamily		Network DASD Storage	OSPARMS OpenSystemServer
RoutingCodeCritical		Miscellaneous	OSPARMS StartupOptions
RoutingCodeNormal		Miscellaneous	OSPARMS StartupOptions
ScratchReuseDelay		VirtualDeviceEngine	VTPARMS DynamicOptions
ScratchVolumesThreshold		VirtualDeviceEngine	OSPARMS DynamicOptions
ShutdownCommands		ParmlibStructure	OSPARMS ParmlibDirectory
StartupCommands		ParmlibStructure	OSPARMS ParmlibDirectory
StartupOptions		ParmlibStructure	OSPARMS ParmlibDirectory
SubAddressSpaceName		Miscellaneous	OSPARMS StartupOptions
Tasklib		Miscellaneous	OSPARMS StartupOptions
UpdateRMFStatistics		Miscellaneous	OSPARMS StartupOptions
VirtualControlUnit		VirtualDeviceEngine	OSDRIVE VirtualDeviceList
VirtualDeviceList		VirtualDeviceEngine	OSPARMS ParmlibDirectory
VolserRange		VirtualDeviceEngine	OSPOOLS PoolSections

Attribute	Feature	Location
VolumePool	VirtualDeviceEngine	OSGROUP GroupSections
VolumePoolDefinitions	ParmlibStructure	OSPARMS ParmlibDirectory
zIIPExploitation	VirtualDeviceEngine	OSPARMS StartupOptions

Parmlib Attributes Related by Feature

The following table groups the attributes by the product feature that uses them. Some product features use the same attributes. These attributes are listed multiple times in the table.

Consider the following:

- An attribute listed with a format of 'xyz | x' indicates the full attribute name followed by its abbreviation.
- The sample Parmlib members can be found in HLQ.CCUUPARM.
- The Location value documents the sample Parmlib member and the section in that member which contains the attribute.

Feature	Attribute	Location
Virtual Volume Compression	HardwareCompressionMethod	OSPARMS Dynamic Options
	HardwareCompressionOption	■ OSPARMS Dynamic Options ■ OSGROUP Group Sections
	MinimumCompressionRate	■ OSPARMS Dynamic Options ■ OSGROUP Group Sections
	MaximumCompressionCPU	■ OSPARMS Dynamic Options ■ OSGROUP Group Sections
Filtering	ExcludeDataSets	OSFILTR Data Set Filters
	Group01-74	OSFILTR Sections
	IncludeDataClass	OSFILTR Data Set Filters
	IncludeDataSets	OSFILTR Data Set Filters
Logger	LogCSASize	OSPARMS Startup Options
	LogDataspaceSize	OSPARMS Startup Options
	LogDetailLevel	OSPARMS Dynamic Options
	LogStream	OSPARMS Dynamic Options

Miscellaneous	Command | Cmd	■ OSPCMDS Shutdown Commands ■ OSSCMDS Startup Commands
	CacheDefaultDataClass	OSPARMS Startup Options
	CacheDefaultStorageClass	OSPARMS Startup Options
	CachePrimary	VTPARMS Dynamic Options
	CacheSecondary	VTPARMS Dynamic Options
	ConsoleCommandTimeout	OSPARMS Dynamic Options
	Description	OSGROUP Group Sections
	DsnameBSDS1	OSPARMS Startup Options
	DsnameLocalVCAT	OSPARMS Startup Options
	DsnameGlobalVCAT	OSPARMS Startup Options
	ForeignTapesExpdt	OSPARMS Dynamic Options
	GlobalReserve	OSPARMS Dynamic Options
	MessagePrefix	OSPARMS Startup Options
	Mode	OSPARMS Architecture
	RoutingCodeCritical	OSPARMS Startup Options
	RoutingCodeNormal	OSPARMS Startup Options
	SubAddressSpaceName	OSPARMS Startup Options
	Tasklib	OSPARMS Startup Options
	UpdateRMFStatistics	OSPARMS Startup Options
Network DASD Storage	DSNameSYSTCPD	OSPARMS OpenSystemServer
	IPAddress	OSPARMS OpenSystemServer
	PipeUnitAddres	OSPARMS OpenSystemServer
	Port	OSPARMS OpenSystemServer
	ProtocolFamily	OSPARMS OpenSystemServer
	TCPName	OSPARMS OpenSystemServer
Parmlib Directory	DatasetFilters	OSPARMS Parmlib Directory
	DynamicOptions	OSPARMS Parmlib Directory
	GroupDefinitions	OSPARMS Parmlib Directory
	ShutdownCommands	OSPARMS Parmlib Directory
	StartupCommands	OSPARMS Parmlib Directory

	StartupOptions	OSPARMS Parmlib Directory
	VirtualDeviceList	OSPARMS Parmlib Directory
	VolumePoolDefinitions	OSPARMS Parmlib Directory
Virtual Device Engine	AllowBTEencryptionForVVE	OSPARMS Dynamic Options
	AllowConcurrentVVEReadAccess	OSPARMS Startup Options
	BypassUCBChecking	OSPARMS Startup Options
	ChpidDeviceList	OSDRIVE Virtual Device List
	IOCpuTimeout	OSPARMS Dynamic Options
	MIHTimeoutValue	OSPARMS Startup Options
	MountRejectThreshold	OSPARMS Dynamic Options
	Offline	OSDRIVE Virtual Device List
	Online	OSDRIVE Virtual Device List
	Pool1-8	OSPOOLS Pool Sections
	PercentRunOnZIIP	OSPARMS Dynamic Options
	ScratchReuseDelay	VTPARMS Dynamic Options
	ScratchVolumesThreshold	OSPARMS Dynamic Options
	VirtualControlUnit	OSDRIVE Virtual Device List
	VolserRange	OSPOOLS Pool Sections
	VolumePool	OSGROUP Group Sections
	zIIPExploitation	OSPARMS Startup Options

OSPARMS Parmlib Member

The SVTSCE JCL procedure for CA Vtape contains a PARMDIR= parameter and an OSPARMS DD for the Parmlib. At startup, the OSPARMS DD data set is searched for the Parmlib member that is named by the PARMDIR parameter (the default is OSPARMS). This member must contain an attribute section named ParmlibDirectory. This section documents which Parmlib members contain the attribute sections required by the product.

The sample OSPARMS member contains the Parmlib Directory, Startup Options, and Dynamic Options sections.

ARCHITECTURE

This section contains the Architecture attributes.

Consider the following:

- This attribute is required. If commented out or not coded, CA Vtape does not start up in the proper mode.
- This section is only loaded when CA Vtape is started.

Mode=

Defines the operating environment and parameter attributes used for CA Vtape.

Valid Values:

- OPENSYS

  Indicates that CA Vtape starts in Linux mode.

- BASIC (default)

  Indicates that CA Vtape starts in z/OS mode.

Required: Yes

Section: <Architecture>

Feature: Miscellaneous

PARMLIB DIRECTORY

This list provides the Parmlib Directory Section attribute names in alphabetic order. The following special values can be used in this section:

- *$THIS* = Indicates that this section is located in the same Parmlib member as the Parmlib Directory Section. If the Parmlib Directory Section is in the member OSPARMS and Startup Commands=$THIS, then the Startup Commands Section is also in the OSPARMS member. By coding $THIS in the Parmlib Directory Section values for the sections that are located in the OSPARMS member, you do not have to change the values when the VTPARMS member is renamed or copied to a member with a different name.
- *$NULL* = Indicates that this section does not exist. This is only a valid value for the Startup Commands, Shutdown Commands, and Volume Pool Definitions Sections.

DatasetFilters=

Valid Values:

- OSFILTR (default)
- Valid PDS member name
- $THIS

Required: Yes

Description: Parmlib member that contains the Dataset Filters Section.

DynamicOptions=

Valid Values:

- OSPARMS (default)
- Valid PDS member name
- $THIS

Required: Yes

Description: Parmlib member that contains the Dynamic Options Section.

GroupDefinitions=

Valid Values:

- OSGROUP (default)
- Valid PDS member name
- $THIS

Required: Yes

Description: Parmlib member that contains the Group Definitions Section.

OpenSystemServer=

Valid Values:

- $THIS (default)
- A valid PDS member name

Required: Yes

Description: Parmlib member containing the Open System Server Section.

StartupCommands=

Valid Values:

- $NULL (default)
- $THIS
- Valid PDS member name

Required: No

Description: Parmlib member that contains the Startup Commands Section.

StartupOptions=

Valid Values:

- OSPARMS (default)
- Valid PDS member name
- $THIS

Required: Yes

Description: Parmlib member that contains the Startup Options Section.

ShutdownCommands=

Valid Values:

- $NULL (default)
- $THIS
- Valid PDS member name

Required: No

Description: Parmlib member that contains the Shutdown Commands Section.

VirtualDeviceList=

Valid Values:

- OSDRIVE (default)
- Valid PDS member name
- $THIS

Required: Yes

Description: Parmlib member that contains the Virtual Device List Section.

VolumePoolDefinitions

Valid Values:

- OSPOOL (default)
- $THIS
- A valid PDS member name

Required: Yes

Description: Parmlib member containing the Volume Pool Definitions Section.

OPEN SYSTEM SERVER OPTIONS

This section contains an alphabetic list of the OpenSystemServer attributes. The list includes a description of each attribute, its valid values, and the CA Vtape features it supports.

Consider the following:

- All of the attributes in this list are required. If commented out or not coded, CA Vtape does not start up in the proper mode.
- Attributes in this section are only loaded when CA Vtape is started.

DsnameSYSTCPD=

Defines the name of the profile data set associated with the TCP/IP subsystem. The profile data set is used to resolve a host name to a TCP/IP address.

Valid Values:

- The TCP/IP subsystem profile data set name.

  The data set name is dynamically allocated at startup with a DD name of SYSTCPD.

- JCL

  The TCP/IP subsystem data set must be coded in the SVTSCE JCL procedure with a DD name of SYSTCPD.

Default: JCL

Required: No

Feature: Miscellaneous

IPaddress=

Defines the TCP/IP address of the CA Cloud Storage for System z Server.

Valid Value: 1 to 48 character TCP/IP address

Default: None

Required: Yes

Feature: Miscellaneous

PipeUnitAddress=

Define a pair of unit address for CTC devices that the local CA Vtape subsystem uses to establish a piped connection with a Linux Server running on System z.

Valid Values: uuu1,uuu2

Two valid unit addresses that are separated by a comma.

Default: There is no default value.

Required: Yes

Feature: Miscellaneous

Port=

Define the port number that the CA Vtape TCP/IP listener uses. Use the attribute to establish connections with the Linux Server.

Valid Values: 1-65535

Default: None

Required: Yes

Feature: Miscellaneous

ProtocolFamily=

Defines the internet protocol family to use when creating sockets for remote connections. The addressing family defines the protocol format of an internet address.

Valid Values:

- IPV4

  Indicates that the IPv4 protocol is used.

- IPV6

  Indicates that the IPv6 protocol is used.

Default: IPV4

Required: No

Feature: MIscellaneous

TCPName=

Defines the one to eight-character TCP/IP address space name that CA Vtape uses. If unspecified, the system derives the value from the TCP/IP configuration file as described in the *IBM z/OS Communications Server: IP Configuration Guide*.

Valid Value: A valid TCP/IP address space name

Default: $NULL

Required: No

Feature: MIscellaneous

STARTUP OPTIONS

This section contains an alphabetic list of the Startup Options attributes. The list includes a description of each attribute, its valid values, and the CA Vtape features it supports.

Consider the following:

- Values in the Valid Values column that are enclosed in quotes must be entered with quotes, for example, DsnameLocalVCAT='CAVTAPE.SYS1.VCAT'
- Values in the Valid Values column that are enclosed in parentheses must be entered with parentheses, for example, RoutingCodeCritical=(1,7,9)
- All of the attributes in this list are optional. If commented out or not coded, the documented default is used.
- Attributes in this section are only loaded when CA Vtape is started. Stop and start CA Vtape for attribute changes to take effect.

AllowConcurrentVVEReadAccess=

This option is available only on JES2 systems.

Enables concurrent read access to Virtual Volumes that CA Disk created. Also enables multiple, simultaneous read access for CA Disk ARCHVOLS on any LPAR.

Valid values:

- NONE
- CADISK

Default: NONE

Feature: Virtual Device Engine

BypassUCBChecking=

Determines if validation is bypassed when updating the Virtual Device UCB's at startup.

Valid Values:

- N

  Bypasses validation.

- Y

  Does not bypass validation.

Default: N

Feature: Virtual Device Engine

CacheDefaultDataClass=

Specifies the SMS data class construct name the system uses to define work data sets dynamically.

Multiple CA Vtape Subsystems that share the Global VCAT also automatically share and reference the same SMS construct names.

Valid Values: Valid DFSMS Data Class name

Set this value with the Data Class that CA Vtape uses to allocate temporary work data sets dynamically.

Note: The value set for this attribute is shared with all CA Vtape Subsystems through the Global VCAT. Any Subsystem starting with a new setting, that is accepted by the operator by replying (Y)es to the SVT*n*I3301I console message, automatically broadcasts the change to all other Subsystems.

Default: NONE

Feature: Miscellaneous

CacheDefaultStorageClass=

Specifies the SMS storage class construct name that CA Vtape uses to define work data sets dynamically.

Multiple CA Vtape Subsystems that share the Global VCAT automatically share and reference the same SMS construct names.

Valid Values: A Valid DFSMS Storage Class name

Set this value with the Storage Class that CA Vtapeuses to allocate work data sets dynamically.

Note: The value set for this attribute is being shared with all other Subsystems through the Global VCAT. Any Subsystem starting with a new setting, that is accepted by the operator by replying (Y)es to the SVT*n*I3301I console message, automatically broadcasts the change to all other Subsystems.

Default: NONE

Feature: Miscellaneous

DsnameBSDS1=

Determines if JCL or dynamic allocation is used to allocate the BSDS1.

Valid Values:

- JCL

  Indicates that the BSDS1 DD in the SVTSCE JCL procedure should be used to allocate the BSDS1 data set.

- The data set name of the BSDS1

  Indicates that dynamic allocation should be used to allocate the BSDS1 data set. If the BSDS1 DD is coded in the SVTSCE JCL procedure, it must point to the same data set name coded for the attribute. An inconsistency generates a critical error aborting startup. Commenting out or deleting the BSDS1 DD in the SVTSCE JCL procedure would be recommended if the BSDS1 data set name is coded for this attribute.

Default: JCL

Feature: Miscellaneous

DsnameGlobalVCAT=

Determines if JCL or dynamic allocation is used to allocate the Global VCAT.

Valid Values:

- JCL

  Indicates that the GLOBAL DD in the SVTSCE JCL procedure should be used to allocate the Global VCAT.

- The data set name of the Global VCAT

  Indicates that dynamic allocation should be used to allocate the Global VCAT. If the GLOBAL DD is coded in the SVTSCE JCL procedure, it must point to the same data set name coded for the attribute. An inconsistency generates a critical error aborting startup. Commenting out or deleting the GLOBAL DD in the SVTSCE JCL procedure would be recommended if the Global VCAT data set name is coded for this attribute.

Default: JCL

Feature: Miscellaneous

DsnameLocalVCAT=

Determines if JCL allocation or dynamic allocation is used to allocate the Local VCAT.

Valid Values:

- JCL

  Indicates that the VCAT DD in the SVTSCE JCL procedure should be used to allocate the Local VCAT.

- The data set name of the Local VCAT

  Indicates that dynamic allocation should be used to allocate the Local VCAT. If the VCAT DD is coded in the SVTS JCL procedure, it must point to the same data set name coded for the attribute. An inconsistency generates a critical error aborting startup. Commenting out or deleting the VCAT DD in the SVTSCE JCL procedure would be recommended if the Local VCAT data set name is coded for this attribute.

  The Local VCAT name can be coded using variables like &SYSID and &SVST, as in the following example:

  ```
  DsnameLocalVCAT='CAVTAPE.&SYSID..&SVTS.VCAT'
  ```

  These variables would be expanded on LPAR PD02 for CA Vtape Subsystem SVT1 to CAVTAPE.PD02.SVT1VCAT.

Default: JCL

Feature: Miscellaneous

LogCSASize=

Specifies the size, in kilobytes, of the ECSA area to store event log records temporarily when the log dataspace cannot be accessed. This area is acquired during CA Vtape initialization. You do not need to change the size unless the logger files are found to be missing records.

Valid Values: 4 to 16

Default: 8

Feature: Logger

LogDataspaceSize=

Specifies the size, in megabytes, of the dataspace that the Internal Logger uses for event log records. A typical loaded environment with LogDetailLevel=1 and the default of 8 MBs holds the last 45-60 minutes of event log records.

Valid Values: 1 to 16

Default: 8

Feature: Logger

MessagePrefix=

Determines the four characters used as a message prefix for all messages that CA Vtape issues.

Note: Do not change the default value unless there is another application using the same message prefix value as CA Vtape.

Valid Values:

- &SVTS

  Translated to the CA Vtape Subsystem number, SVT*n*, where *n* is 1-8, depending on which Subsystem issues the message. For example, all messages issued by Subsystem SVT1 start with SVT1 as in SVT1R0200E. All messages issued by Subsystem SVT2 start with SVT2 as in SVT2R0200E.

- Any 4 character value or variable

  If you enter four characters (for example SVTS) all messages that the Subsystems using this Parmlib issue, start with SVTS as in SVTSR0200E.

Default: &SVTS

Feature: Miscellaneous

MIHTimeoutValue=

Specifies the Missing Interrupt Handler (MIH) timeout value in seconds. CA Vtape issues SETIOS commands to set the MIH timeout for its Virtual Devices to this value at startup. Upon shutdown, the system issues SETIOS commands to set the MIH value to two seconds for the now unusable Virtual Devices.

We recommend a setting of 30 seconds.

Do not set this attribute to a value higher than 60 seconds. Contact CA Support.

If you do not want CA Vtape to set the MIH timeout value for its Virtual Devices and have a process outside of CA Vtape to set the value, set this attribute to a value of -1. When MIHTimoutValue is set to a value of -1, CA Vtape does not set the MIH timeout.

Important! If you set this attribute to a value of -1 and you do not use a process outside of CA Vtape to set the MIH timeout for the Virtual Devices to an appropriate value or set an inappropriate value, you could provoke mount pending situations and IOS errors. This could result in the Virtual Devices being fenced or boxed which can require an IPL to correct.

Valid Values: -1 to 9999

Default: 15

Feature: Virtual Device Engine

RoutingCodeCritical=

Determines the route code or codes that CA Vtape uses for high priority console messages.

Valid Values:

- -1

  Indicates that a route code of 1 (master console) should be used.

- A single valid route code

  Indicates the single route code to use for high priority console messages.

- (code1,code2,…)

  Indicates the multiple route codes to use for high priority console messages.

Default: -1

Feature: Miscellaneous

RoutingCodeNormal=

Determines the route code or codes that CA Vtape uses for normal priority console messages.

Valid Values:

- -1

  Indicates that the ROUTCODE keyword value on the DEFAULT statement in the CONSOL*xx* member of SYS1.PARMLIB or *SYSn*.IPLPARM should be used.

- A single valid route code

  Indicates the single route code to use for normal priority console messages.

- (code1,code2, …)

  Indicates the multiple route codes to use for normal priority console messages.

Default: -1

Feature: Miscellaneous

SubAddressSpaceName=

Specifies the JCL procedure name to use when starting the sub address spaces for Virtual Control Units.

Valid Values:

- A valid JCL procedure library member name.

  Indicates that the sub address spaces start with a copy of the SVTSCE JCL procedure which has been copied to a member with this name.

- &JOBNAME

  Indicates that the sub address spaces start with the same JCL procedure the main task was started with.

Default: &JOBNAME

Feature: Miscellaneous

Tasklib=

Determines if Split Maintenance-Level Protection is off or running in Steplib, Automatic, or Library mode. Split Maintenance-Level Protection prevents the CA Vtape address spaces from being started or restarted with different maintenance levels and issues warning messages when mismatched maintenance levels are detected.

Valid Values:

- Disabled

  Turns off Split Maintenance-Level Protection. The system loads the runtime modules from the STEPLIB DD or a link listed data set and is not validated.

- Steplib

  This mode activates Split Maintenance-Level Protection and loads the run-time modules from the STEPLIB DDs in the started task JCL procedures or a link-listed loadlib. If a Virtual Control Unit is restarted, the run-time modules are loaded from the STEPLIB DD or a link-listed loadlib. If the modules loaded do not match the maintenance level of the modules the other sub address spaces use, an error report is produced and repeated every 60 seconds.

- Automatic

  This mode activates Split Maintenance-Level Protection and is the recommended setting. In this mode, a temporary loadlib or Tasklib is dynamically allocated using the name *prefix*.D*xxxxxx*.T*yyyyyy*.LIB*n* where *prefix* is the CA Vtape data set prefix, *xxxxxx* is the creation date, *yyyyyy* is the creation time, and *n* is the last digit of the Subsystem number (1-8). The temporary loadlib is dynamically APF-authorized and loaded with the modules found in the SVTSCE JCL procedure STEPLIB DD. Dynamic APF authorization requires that CA Vtape have UPDATE access to the FACILITY class CSVAPF if this is secured. The temporary loadlib is then used to start all the CA Vtape started tasks in this Subsystem. If a Virtual Control Unit is restarted, the temporary loadlib is used. The STEPLIB DD loadlibs and link-listed loadlibs are ignored. The temporary loadlib is deleted when CA Vtape is shut down.

  This mode allows maintenance to be applied to a loadlib referred to by the STEPLIB DD or in the link-list without any impact while CA Vtape is running. Stop and start the entire Subsystem so that the changed modules load.

- An existing loadlib data set name

  Known as library mode, this setting activates Split Maintenance-Level Protection. It is the same as Automatic mode except that the loadlib data set name coded as the attribute value is used in place of a dynamically allocated loadlib. The loadlib data set name must contain at least two nodes and must not be present in the STEPLIB DD in the SVTSCE JCL procedure. If the loadlib is not APF authorized, it will be dynamically APF authorized by CA Vtape. Dynamic APF authorization requires that CA Vtape have UPDATE access to the FACILITY class CSVAPF if this is secured.

  Start the data set name with the CA Vtape DSN prefix to avoid more security authorizations.

Note: When running in Automatic or Library mode, if a startup error occurs with allocating, accessing, loading, or dynamically APF-authorizing the loadlib, CA Vtape issues error messages and terminates the startup.

Default: Automatic

Feature: Miscellaneous

UpdateRMFStatistics=

Determines if CA Vtape updates the Channel Measurement Block (CMB) of the Virtual Devices. This information is collected and reported on by the Resource Measurement Facility (RMF).

Valid Values:

- Y

  CA Vtape updates the CMB.

- N

  CA Vtape does not update the CMB.

Default: Y

Feature: Miscellaneous

zIIPExploitation=

Determines if CA Vtape is to use the zIIP processor. This attribute should only be specified if your system supports the zIIP processor (z9 and later CPUs with the required level of z/OS). A companion parameter in the DynamicOptions Section is required to specify what percentage of CA Vtape work is eligible to run on the zIIP, see the PercentRunOnZIIP= attribute in this chapter.

Valid Values:

- Y

  Indicates that CA Vtape is to use the zIIP processor. Also specify a value greater than 0 for the PercentRunOnZIIP attribute in the Dynamic Options member to cause work to be made eligible for the zIIP. You can specify this value even if you do not have a zIIP processor installed. When you run in a z9 or later CPUx without a zIIP processor installed, you can specify this value and PercentRunOnZIIP = 100 to simulate zIIP eligibility for capacity planning purposes. For more information about monitoring and performing capacity planning for zIIP processors, see "Performance."

- N

  Indicates that CA Vtape does not use the zIIP processor.

Default: N

Feature: Virtual Device Engine

DYNAMIC OPTIONS

This section contains an alphabetic list of the Dynamic Options attributes. The list includes a description of each attribute, its valid values, and the CA Vtape features it supports.

While active, the CA Vtape subsystem can dynamically reload the Dynamic Options attributes by using the following operator command:

```
SVTn REFRESH=OPTION
```

This is an alphabetic list of the Dynamic Options attributes:

AllowBTEencryptionForVVE=

Determines when Virtual Volumes are eligible for encryption by CA Tape Encryption. CA Tape Encryption is a separately licensed product.

Valid Values:

- N

  Virtual Volumes are not eligible for encryption.

- Y

  Virtual Volumes are eligible for encryption.

Default: Y

Feature: Virtual Device Engine

CachePrimary=

Specifies the Primary space allocation, in MB, used for dynamically allocated Virtual Volume data sets.

The recommended value for this attribute is 1/10th of the Virtual Volume size, for example: 40, 80, or 200.

Valid Values: 24 to 2000

Default: 200

Feature: Miscellaneous

CacheSecondary=

Specifies the Secondary space allocation, in MB, used for dynamically allocated Virtual Volume data sets.

The recommended value for this attribute is 1/10th of the Virtual Volume size, for example: 40, 80, or 200.

Valid Values: 24 to 2000

Default: 200

Feature: Miscellaneous

ConsoleCommandTimeout=

Controls the number of seconds before console commands timeout and WTOR SVT*n*D9920I is issued.

Note: We recommend a setting of 60, which allows 60 seconds for a CA Vtape console command to be executed before the WTOR is issued.

Valid Values: 30 to 600

Default: 300

Feature: Miscellaneous

DefaultGroup=

Controls the group number that is assigned to a Virtual Volume when the data set cannot be identified based on data class or data set name filter. An example is a Virtual Volume that is mounted on Virtual Device by a specific unit allocation:

```
//DD1 DD UNIT=/3455,DISP=(NEW,KEEP),..
```

Valid Values: GROUP01, GROUP02, GROUP03,… to GROUP74

Default: GROUP01

Feature: Miscellaneous

ForeignTapesExpdt=

Determines the expiration date value that causes CA Vtape to bypass a mount request for a VOLSER that matches a Virtual Volume. For example, assume that your Virtual Volume range is V00000-V99999 and a vendor ships you a tape with a VOLSER of V15000. If a job is run to read the tape using an esoteric that includes the CA Vtape virtual devices, CA Vtape tries to intercept the mount. To bypass or ignore the mount request, add the LABEL=EXPDT=*value* to the JCL DD statement. *value* is the ForeignTapesExpdt attribute value.

Note: Changing the DD statement UNIT parameter to an esoteric or generic that does not include the CA Vtape virtual devices also causes CA Vtape to bypass or ignore the mount request.

Valid Values:

- 00000 or '1900/000'

  Indicates that the feature is turned off or disabled.

- A valid date in yyddd or 'yyyy/ddd' format that is not a never-expire date

  Indicates the date value. 99365, 99366, '1999/365', and '1999/366' (never-expire dates) are all reserved values that will generate a validation error if used for this attribute.

Default: 00000 or '1900/000'

Feature: Miscellaneous

GlobalReserve=

Determines whether the Global VCAT reserve is issued with a QNAME.RNAME of “SVTS.GLOBAL” or “SVTS.Global VCAT data set name”.

Valid Values:

- Compatibility

  Indicates that GLOBAL is used for the reserve RNAME. This allows backward compatibility with subsystems running at Release 2.0 SP05 or prior maintenance.

  Important! The use of this generic enqueue reserve RNAME instead of the actual Global VCAT name, causes contention and a performance penalty between CA Vtape complexes. When a complex issues the generic reserve to update its Global VCAT, other complexes wait to issue the reserve to update their Global VCAT. Even though the different complexes are using different Global VCATs, updates are serialized across all complexes.

- Enhanced

  Indicates that the Global VCAT names that are used for the reserve RNAME. This prevents contention between multiple CA Vtape Complexes, by issuing the Global VCAT reserve with a reserve name unique to the Global VCAT. In this mode, all subsystems in all CA Vtape Complexes must be running in enhanced mode. If a back-leveled or compatibility mode subsystem is detected, enhanced mode subsystems issues messages and change to compatibility mode.

Default: Enhanced

Feature: Miscellaneous

HardwareCompressionMethod=

Determines which compression methods can be employed when compression is active and compression is used for data that is written to a Virtual Volume.

Valid Values:

- $NULL

  $NULL is a compatibility setting that causes this attribute to be ignored. If compression is activated and this attribute is set to $NULL, CA Vtape utilizes RLL and CMPSC compression. LZ78 is not employed.

- CMPSC

  CMPSC is the hardware compression call instruction. This method employs several compression dictionaries to achieve moderate compression rates with less CPU overhead than LZ78.

- RLL

  RLL (Run Length Limited) is a software implementation which compresses data by eliminating repeating characters. This method can sometimes achieve good compression with low CPU overhead.

- LZ78

  LZ78 (Lepel-Ziv-78) is a software implementation. This method utilizes an adaptive compression algorithm which can achieve higher compression rate but at much higher CPU rates than CMPSC or RLL.

- ALL

  ALL indicates that all of the compression methods mentioned above except $NULL should be used. That is, CMPSC, RLL, and LZ78 are all used.

Note: More than one compression method can be specified for HardwareCompressionMethod by separating each method using a comma.

For example:

```
HardwareCompressionMethod= CMPSC,RLL
HardwareCompressionMethod= LZ78,RLL
```

When more than one compression method is employed, CA Vtape automatically monitors each method and chooses the method achieving the highest compression rate for a given Virtual Volume.

Default: ALL

Feature: Virtual Volume Compression

HardwareCompressionOption=

Determines if compression is performed on data that is written to Virtual Volumes. If the network storage being used already performs compression, this option should be set to N.

Valid Values:

- N

  Indicates that compression is never be performed.

- Y

  Indicates that compression is always be performed.

- JCL

  Indicates that compression is erformed if TRTCH=COMP is coded in the JCL or if COMPACT=YES was used when defining the Virtual Devices. If TRTCH=NOCOMP is coded, compression is not performed.

- GROUP

  Indicates that the Group Definitions are checked for one of the above three values.

Note: For more information about Virtual Volume Compression, see the chapter "Virtual Device Engine."

Default: N

Feature: Virtual Volume Compression

IOCpuTimeout=

Determines the number of CPU seconds allowed for each I/O before the operating system terminates the I/O with ABEND code S05B. The timeout prevents the Virtual Device Engine from looping.

Valid Values: 5 to 300

Default: 15

Feature: Virtual Device Engine

LogDetailLevel=

Determines the level of detail of internal processes that the Logger records.

Valid Values:

- 0

  Indicates that no logging should be done.

- 1

  Indicates that basic events like messages issued, console commands issued, batch utilities executed, recovery errors, subtask attach/detach, and major routines entered should be logged. We recommend this setting unless you need more log detail to diagnose a problem.

- 2

  Indicates that in addition to level 1 details, subchannel commands, interrupt commands, and low-level routines entered are logged.

- 3

  Indicates that in addition to level 2 details, detailed I/O information are logged.

LogDetailLevel=2 and 3 may generate a large volume of records on an LPAR with a very active Subsystem.

HLQ.CCUUJCL(LOGEVENT) contains complete descriptions of what events are logged and the detail level they are logged at.

Default: 1

Feature: Logger

LogStream=

Determines whether the data generated by the Logger is copied from the CA Vtape log dataspace to an IBM system Log Stream data set. This is referred to as the External Logger.

Valid Values:

- NONE

  Indicates that a Log Stream does not exist so a copy does not occur. The Logger overwrites the data that is generated previously in the log dataspace.

- 'A valid Log Stream name'

  Indicates that a Log Stream does exist. As each block in the log dataspace is filled, a call is made to the System Logger to write the block to the defined Log Stream. If errors are detected, the System Logger and CA Vtape issues messages and this attribute is internally changed to a value of NONE.

Note: The only valid Log Stream name is created using the IBM IXCMIAPU utility. Use the SVT*n* Display Log console command to determine the status of the logger.

Default: NONE

Feature: Logger

MaximumCompressionCPU=

Determines the maximum amount of data that is compressed in each 50 MB segment of data that is written to a Virtual Volume. This percentage is directly related to the amount of CPU used by compression. If, for example, this percentage is set to 75, only 75% of the data written will be compressed and only 75% of the CPU that would have been used by compression is actually used.

Valid Values:

- 0

  Indicates that no data is compressed. Use this setting temporarily. If compression should be turned off permanently, use the HardwareCompressionOption attribute.

- 1 to 99

  Indicates the actual percentage of data written that is compressed. If set to 50, only 50% of each 50 MBs of data written to a Virtual Volume will be compressed.

- 100

  Indicates that all the data written data is compressed.

Note: For more information about Virtual Volume Compression, see the chapter “Virtual Device Engine.”

Default: 100

Feature: Virtual Volume Compression

MinimumCompressionRate=

Determines the minimum compression percentage that must be achieved to continue compressing a 50 MB segment of a Virtual Volume being written. Compression is tested every 50 MBs of each Virtual Volume to determine the best compression method to use.

Valid Values:

- 0

  Indicates that this verification is disabled. Any achieved percentage compression or no compression allows compression to continue.

- 1 to 100

  Indicates the percent of compression that must be achieved to continue compressing each 50 MB segment of a Virtual Volume.

Note: For more information about Virtual Volume Compression, see the chapter "Virtual Device Engine."

Feature: Virtual Volume Compression

MountRejectThreshold=

Determines the action that the system takes when the tape management system rejects a scratch mount.

Valid Values:

- IGNORE

  Indicates that no action is taken. CA Vtape continues selecting Virtual VOLSERs from the scratch pool until the tape management accepts a VOLSER or the pool is empty. This setting can lead to a scratch shortage.

- WTOR,*nn*,

  Where *nn* is between 05 and 99, indicates that after *nn* VOLSERs are rejected for this mount request a WTOR is issued. The resulting SVT*n*V1803W message asks the operator whether the job is canceled or whether the Mount Reject Policy is ignored.

- Cancel,*nn*,

  Where *nn* is between 05 and 99, indicates that after *nn* VOLSERs are rejected for this mount request the job is canceled.

Note: Cancel,10 is the recommended setting.

Default: IGNORE

Feature: Virtual Device Engine

PercentRunOnZIIP=

Specifies the percentage of work to be made eligible for the zIIP processor. In the Startup Options Section, attribute zIIPExploitation= must be set to Y to cause PercentRunOnZIIP to take effect. If PercentRunOnZIIP= is nonzero but zIIPExploitation=N is specified an error message will be issued and CA Vtape will run without zIIP processing activated.

Valid Values:

- 0

  Specifies that no work is directed to the zIIP processor. You can specify PercentRunOnZIIP=0 and zIIPExploitation=Y simultaneously. In this case, CA Vtape issues the warning message SVT*n*I0009W but builds and maintains the internal resources required to use the zIIP processor. However, no work is made zIIP-eligible until a nonzero value is specified and the dynamic options are refreshed using the SVT*n* REFRESH=OPTIONS command.

- 1 to 100

  Specifies the percentage of work made eligible for the zIIP processor. We recommend a value of 100 to take full advantage of the zIIP processor. You can use IBM's WLM to prioritize which products are eligible for the zIIP processor.

Default: 100

Feature: Virtual Device Engine

RealStorageSafetyThreshold=

Specifies the percentage which the z/OS available central storage satisfactory threshold is multiplied by to determine the point at which CA Vtape defers mount processing to help z/OS maintain a healthy central storage environment.

z/OS maintains two fields which are used to determine the health of its central or real storage. The fields are RCEAFCLO and RCEAFCOK in the RCE control block. RCEAFCLO is the available central storage low threshold. Reaching this point indicates to z/OS that aggressive actions must be automatically taken to prevent a real storage constraint from impacting the system. RCEAFCOK is the available central storage satisfactory threshold or safety point. Reaching this point indicates to z/OS that aggressive actions are no longer required because the real storage constraint has been relieved.

CA Vtape dynamically acquires real storage for its active virtual devices. As this activity increases, especially in a larger installation with many virtual devices used concurrently, the amount of real storage usage can cause the available real storage to drop below the safety point. The RealStorageSafetyThreshold allows you to tell CA Vtape when to stop acquiring real storage to ensure that CA Vtape does not cause real storage usage to go lower than the safety point.

For example, the default of 400% implies that CA Vtape keeps a mount in pending status if the storage that needs to be acquired reduces the available real storage to within 400% of the safety point. As other tasks that do not control themselves might be requesting real storage simultaneously, the default of 400% is a conservative value to delay CA Vtape real storage requests well before z/OS experiences real storage stress. If you see CA Vtape delaying mounts on a system where a storage constraint is not occurring, then lower the default value.

If the attribute value is set too high, CA Vtape issues the SVT*n*V5008I message delaying new Virtual Mount requests and appears to hang. The system continues to process previous mounts. Only new requests are delayed. When previous requests complete and free their storage, a new Virtual Mount request starts unless another task consumes the freed storage. Lowering the attribute value and issuing the SVT*n* REFRESH=OPTIONS console command for each CA Vtape subsystem that picks up the change eliminates the delays.

To disable this feature, specify a value of 0.

Valid Values: 0 to 900

Default: 400

Feature: Miscellaneous

ScratchReuseDelay=

This attribute Influences the number of days to delay the reuse of a scratched virtual volume. For example, do consider a virtual volume scratched today also satisfies a new scratch mount request until (*n*) days have elapsed.

If the attribute value cannot be honored because no *n* day-old scratches are available, the subsystem takes the following actions:

- The current mount request is satisfied by selecting the first available scratch volume regardless of age.
- The age in days of the oldest scratch volume that is found within the pool is used to override the attribute for one hour.
- An SVT*n*S7023I message is issued describing that the override is in effect for this pool.

Valid Values: 0 to 128

Default:5

Feature: Virtual Device Engine

ScratchVolumesThreshold=

Determines whether an SVT*n*D0928W message is issued every 30 minutes when the number of available scratch Virtual Volumes drops below the specified value.

Valid Values:

- 0

  Indicates that the available scratch Virtual Volumes are not being monitored. The warning message is never issued.

- 1 to 500000

  Indicates the scratch Virtual Volume threshold value. When the number of scratch Virtual Volumes falls below the specified value, the warning message is issued. The message is reissued every 30 minutes as long as the number of scratch Virtual Volumes remains below the threshold value.

Default: 0

Feature: Virtual Device Engine

OSDRIVE Parmlib Member

OSDRIVE is a Parmlib member containing Virtual Control Unit numbers and device addresses used by CA Vtape to start Virtual Control Units and Virtual Devices. The Virtual Devices are local in scope, unique to a system or LPAR. They are affectively "cabled" to the Virtual Control Units they are defined to. Virtual Device 100 on LPAR SYS1 is not the same device as Virtual Device 100 on LPAR SYS2. Both devices can be on-line and servicing different tape mounts at the same time.

This section contains an alphabetic list of the Virtual Drive List attributes. The list contains a description of each attribute, its valid values, and the CA Vtape features it supports.

ChpidDeviceList=

A valid online Channel Path ID (CHPID) must be assigned to every Virtual Device in order to successfully vary them online. ChpidDeviceList is an optional attribute which can be used to define up to eight physical devices from which active CHPIDs will be assigned to the devices of a Virtual Control Unit.

When the ChpidDevList attribute is not specified, the Subsystem will look for the device assigned to the //CHPID DD in the SVTSCE JCL procedure and it will use those CHPIDS for all the devices defined by all Virtual Control Units. If the //CHPID DD is not present, the Subsystem will use the CHPIDs assigned to the DASD device associated with the BSDS1 control data set.

This attribute offers the greatest degree of flexibility since it allows you to assign different CHPIDs to different Virtual Control Units to eliminate a single point of failure.

Up to eight, comma delimited devices can be defined. Only the first 8 CHPIDs found when parsing the device list will be used.

Consider the following:

- CHPIDs that are initially assigned to a Virtual Device remain in effect until the next IPL. If you change the ChpidDeviceList, CHPID DD or BSDS1 location and then restart the Subsystem, the modifications will have no effect on the CHPIDs used for previously defined Virtual Devices. The modifications will take effect only for newly defined devices.
- Our recommendation is to define two Virtual Control Units with different ChpidDeviceList values. Those values should include at least two non-DASD devices that utilize different CHPIDs. This will ensure a minimum of two non-DASD CHPIDs are assigned to each Virtual Device and no more than half of the Virtual Devices are using these same CHPIDs.

Valid Values:

- uuuu

  Defines a single Device address.

- uuu0,uuu1,…,uuux

  Defines a list of individual device addresses separated by commas.

- uuu0-uuux | uuu0:uuux

  Defines a range of device addresses with the starting and ending address separated by a dash (-) or a colon (:).

- uuu0,uuu1,uuu2-uuux

  Defines a list and a range of device addresses separated by commas.

Feature: Virtual Device Engine

Offline=

Determines which Virtual Devices are assigned to a particular VirtualControlUnit and that these devices should not be varied online when CA Vtape is started. The attribute can be repeated as many times as needed to define all the Virtual Devices for a Control Unit.

Valid Values:

- uuuu

  Defines a single Virtual Device address.

- uuu0,uuu1,…,uuux

  Defines a list of individual Virtual Device addresses separated by commas.

- uuu0-uuux | uuu0:uuux

  Defines a range of Virtual Device addresses with the starting and ending address separated by a dash (-) or a colon (:).

- uuu0,uuu1,uuu2-uuux

  Defines a list and a range of Virtual Device addresses separated by commas.

Feature: Virtual Device Engine

Online=

Determines which Virtual Devices are assigned to a particular VirtualControlUnit and that these devices should be varied online when CA Vtape is started. The attribute can be repeated as many times as needed to define all the Virtual Devices for a Control Unit.

Valid Values:

- uuuu

  Defines a single Virtual Device address.

- uuu0,uuu1,…,uuux

  Defines a list of individual Virtual Device addresses separated by commas.

- uuu0-uuux | uuu0:uuux

  Defines a range of Virtual Device addresses with the starting and ending address separated by a dash (-) or a colon (:).

- uuu0,uuu1,uuu2-uuux

  Defines a list and a range of Virtual Device addresses separated by commas.

Feature: Virtual Device Engine

VirtualControlUnit=

Determines the number of Virtual Control Unit address spaces that are started. Up to eight can be started. Each address space controls only those Virtual Devices defined to it.

Note: We recommend that you define at least two Virtual Control Units. This will allow one Control Unit to be restarted if it develops a problem, affecting only its Virtual Devices while the other Control Unit and its devices continue to service virtual tape mounts.

Valid Values: 1 to 8

Default: 1

Feature: Virtual Device Engine

OSGROUP Parmlib Member

The OSGROUP Parmlib member contains group attribute definitions. Groups define policies that are applied to a Virtual Volumes subset. Group attributes control certain behaviors. For example, if compression is used or which VOLSERs pool a scratch tape is chosen from to service the tape mount.

While active, the Subsystem can dynamically reload the group member attributes by using the SVT*n* REFRESH=GROUP console command.

Because the group definitions are local definitions that are Subsystem-specific, issue the SVT*n* REFRESH=GROUP console command for each Subsystem that shares a common modified group member. Otherwise, only the Subsystem for which the command was issued picks up the changes.

OSGROUP contains the index section Group Definitions, which references one or more group sections with customizable names.

Group Definitions

The Group Definitions section functions as an index to your customized Group Sections. It contains a complete list of all the Groups.

All Groups can point to the same Group Section. All Groups can point to different Group Sections. Any combination of Groups can use the same Group Section or a unique Group Section. The only rules are that all Groups must be coded and a Group cannot point to a Group Section that does not exist.

You can customize the coded value for each Group attribute to any name that allows you to identify the Group purpose. The sample names associate a mount point on Linux on System z with a group.

Code the customized Group Section name inside less than and greater than signs (<*Customized_Group_Section_Name*>) in the same Parmlib member as the Group Definitions Section.

The valid values and description of the Group Definitions Group*nn*= attribute is as follows:

Group*nn*=

Assigns a set of attributes, which a customizable section name defines, to a group. Each of the 32 groups (Group01 - Group04, Group11 - Group14, …, Group71 - Group74) must be defined. The assigned value must be a section name that is defined in the same Parmlib member.

Valid Values: A defined Group Section

Feature: Miscellaneous

Group Sections

This section contains an alphabetic list of the attributes that can be coded in the customizable Group Sections. The list contains a description of each attribute, its valid values, and the CA Vtape features it supports. Unless noted, each attribute is local in scope.

Description=

A free form group description that is limited to 30 characters. Enclose the coded value in single quotes.

Valid Values: 'Any 30 characters'

Feature: Miscellaneous

HardwareCompressionOption=

Determines if compression is performed on data written to Virtual Volumes that are assigned to this group.

Note: This attribute is only active if the HardwareCompressionOption attribute in the Dynamic Options Section is set to GROUP.

Valid Values:

- N

  Indicates that compression is never be performed.

- Y

  Indicates that compression is always be performed.

- JCL

  Indicates that compression is be performed if TRTCH=COMP is coded in the JCL or if COMPACT=YES was used when defining the Virtual Devices. If TRTCH=NOCOMP is coded in the JCL, compression is not performed.

Note: For more information about Virtual Volume Compression, see the chapter "Virtual Device Engine."

Default: N

Feature: Virtual Volume Compression

MaximumCompressionCPU=

Determines the maximum data amount in each 50MB segment of data that is written to a Virtual Volume to be compressed. This percentage is directly related to the CPU amount the compression uses. For example, if you enter 75, then the system compresses 75% of the data that is written and uses 75% of the CPU that compression would have required.

Note: To activate this attribute, in the Dynamic Options Section, set the HardwareCompressionOption attribute to GROUP.

Valid Values:

- 0

  Indicates that no data is compressed. Use this setting temporarily. If you want compression turned off permanently, configure the HardwareCompressionOption attribute.

- 1 to 99

  Indicates the actual percentage of data written that is compressed. If you enter 50, then 50% of each 50MBs of data that is written to a Virtual Volume is compressed.

- 100

  Indicates compress all the data.

Note: For more information about Virtual Volume Compression, see the chapter "Virtual Device Engine."

Default: 100

Feature: Virtual Volume Compression

MinimumCompressionRate=

Determines the minimum compression percentage that must be achieved to continue compressing a 50MB segment of a Virtual Volume being written. Compression is tested every 50 MBs of each Virtual Volume to determine the best compression method to use.

Note: This attribute is only active if the HardwareCompressionOption attribute in the Dynamic Options Section is set to GROUP.

Note: For more information about Virtual Volume Compression, see the chapter "Virtual Device Engine."

Valid Values: 1 to 100

Section: Group Sections

Feature: Virtual Volume Compression

VolumePool=

Specifies the name of the Virtual Volume pool that is used to obtain the volser for new scratch mount requests.

Virtual Volume pools are defined in the Volume Pool Definitions Section of the OSPOOLS member.

Valid Values: POOL1 or any valid virtual volume pool that is defined under the Volume Pool Definitions Section.

Default: POOL1

Section: Group Sections

Feature: Miscellaneous

OSFILTR Parmlib Member

The OSFILTR Parmlib member contains Data Set Name Filter include/exclude definitions and SMS Data Class include definitions. These definitions determine the data sets that CA Vtape processes, and the groups that the system assigns to those data sets.

Important! Define at least one Include section containing one filter for CA Vtape to intercept tape mounts.

While active, the CA Vtape Subsystem can dynamically reload the filter member attributes by using the SVT*n* REFRESH=FiLTers console command.

Because the filter member contains local definitions that are Subsystem-specific, issue the SVT*n* REFRESH=FiLTers console command for each Subsystem that shares a common modified filter member. Otherwise, only the Subsystem for which the command was issued picks up the changes.

OSFILTR contains an index section that is named Dataset Filters. This section references at least one include data set or include data class section with customizable names. Multiple include data set, exclude data set, and include data class sections can be referenced. Each customized section name that is referenced must be unique and must exist in the same member as the Dataset Filters Section. A section that does not exist provokes a validation error when CA Vtape is started or when the SVT*n* REFRESH=FILTERS console command is issued.

Data Set Filters Section

This section contains an alphabetic list of the Data Set Filters attributes. The list contains a description of each attribute, its valid values, and the CA Vtape features it supports.

ExcludeDatasets= | ExcludeDSN= | ExcludeDS=

Determines the customizable section name of a section containing data set name exclude entries. The value coded must match a section name enclosed in greater than and less than signs in the same Parmlib member as the Data Set Filters section itself. Multiple exclude data set sections can be defined and identified by multiple ExcludeDataSets attributes. Each section must have a unique name.

For example:

```
<DatasetFilters>
ExcludeDatasets= 'Prod_DataSet_Excludes'
ExcludeDS= 'Test_DataSet_Excludes'
```

Note: If no data set name *exclude* entries are needed, the attribute can be commented out or deleted or the attribute can point to a valid section that contains no entries.

Valid Values: A valid section name

Default: EXCLUDEDATASETS

Feature: Filtering

IncludeDatasets= | IncludeDSN= | IncludeDS= |

Determines the customizable section name of a section containing data set name include entries. The value coded must match a section name enclosed in greater than and less than signs in the same Parmlib member as the Data Set Filters section itself. Multiple include data set sections can be defined and identified by multiple IncludeDataSets attributes. Each section must have a unique name.

For example:

```
<DatasetFilters>
IncludeDatasets= 'Prod_DataSet_Includes'
IncludeDS= 'Test_DataSet_Includes'
```

Note: If no data set name *include* entries are needed, the attribute can be commented out or deleted or the attribute can point to a valid section that contains no entries.

Valid Values: A valid section name

Default: INCLUDEDATASETS

Feature: Filtering

IncludeDataClass= | IncludeDC=

Determines the customizable section name of a section containing SMS data class entries. The value coded must match a section name enclosed in greater than and less than signs in the same Parmlib member as the Data Set Filters section itself. Multiple include data class sections can be defined and identified by multiple IncludeDataClass attributes. Each section must have a unique name.

For example:

```
<DatasetFilters>
IncludeDataClass  = 'Prod_Data_Class_Includes'
IncludeDC         = 'Test_Data_Class_Includes'
```

Note: If no data class include entries are needed, the attribute can be commented out or deleted or the attribute can point to a valid section that contains no entries.

Valid Values: A valid section name

Default: INCLUDEDATACLASS

Feature: Filtering

Include Data Class Section

This section contains the description of the Group attribute and its valid value.

Group*nn*=

Indicates that a tape mount for a data set, with this SMS data class assigned to it, should be intercepted and written to Virtual Volumes. Also determines which group (01-04, 11-14…, or 71-74) is assigned to the Virtual Volumes mounted.

The SMS data class value used must be defined to SMS. The value used must be the full data class name. A wildcard pattern is not allowed.

Valid Values: A valid SMS data class

Feature: Filtering

Include Data Sets Section

This section contains the description of the Group attribute and its valid values.

Group*nn*=

Indicates that a tape mount for a data set which matches the data set name or the pattern value is intercepted and written to Virtual Volumes. Also determines which group (01-04, 11-14..., or 71-74) is assigned to the Virtual Volumes mounted.

Valid Values:

- 'A valid data set name'
- 'A valid data set name pattern'

  Note: For more information about coding data set name patterns, see the chapter "The Tape Mount Intercept Filters."

Feature: Filtering

Exclude Data Sets Section

This section contains the description of the Group attribute and its valid values.

Group*nn*=

Indicates that a tape mount for a data set which matches the data set name or pattern value should not be intercepted and written to CA Vtape Virtual Volumes if it was intercepted by an include data set name entry for the same group.

For example, an include data set name entry of Group13='PROD./' would intercept a tape mount for a data set named PROD.PAYROLL and assign it to group 13. An exclude data set name entry of Group13='PROD.PAYROLL' would indicate to CA Vtape that PROD.PAYROLL should not be intercepted and assigned to group 13. PROD.PAYROLL could be intercepted by another filter and assigned to a group other than 13.

Valid Values:

- 'A valid data set name'
- 'A valid data set name pattern'

  Note: For more information about coding data set name patterns, see the chapter "The Tape Mount Intercept Filters."

Feature: Filtering

OSPOOLS Parmlib Member

OSPOOLS is a Parmlib member containing the Volume Pool Definitions section.

Volume Pool Definitions Section

This section describes attributes that you use to define volume pools and VOLSER ranges that CA Vtape uses. You can define up to eight volume pools.

Volume pools are used to select the VOLSER assigned to a virtual scratch mount. When a scratch mount occurs, filters defined in the OSFILTR member assign a group number to the mount. The Group Section assigns a pool name that is defined in the OSPOOLS member. The Pool Section contains a VOLSER range from which a VOLSER is chosen to satisfy the mount.

The volume pool definitions can be dynamically reloaded by using the SVT*n* REFRESH=POOLS console command.

When this command is issued, CA Vtape dynamically adds and deletes ranges as appropriate using the latest attributes defined within the OSPOOLS member. The volume pool definitions are global in scope, therefore all Subsystems sharing the control files are modified when this command is issued.

This section serves as an index to the user that is defined pool sections. The attributes associate a user that is defined pool section with a specific pool.

POOL*n*=

This section contains attributes which define volser ranges that are associated with a given pool of virtual VOLSERs.

Valid Values: A valid user that is defined section name. The value that you enter must match a section name that is enclosed in greater than and less than signs. Also, the section must be in the same Parmlib member as the Volume Pools Definition Section itself.

Section: <VolumePoolDefinitions>

Feature: Volume Pooling

VolserRange=

Defines a virtual VOLSER range within the pool. Specify as many ranges as necessary to define the total number of Virtual Volumes within the pool. The first non-matching character of the VOLSER range is the Virtual Volume prefix. The first character is always considered a prefix character. The suffix characters must be '00' in the low VOLSER of the range and '99' in the high VOLSER of the range.

Valid Values:

- *vvvv*00-*vvvv*99

  Defines the minimum range of 100 VOLSERs where *vvvv* would be a unique prefix value.

- *vvv*000-*vvv*999

  Defines a range of 1000 VOLSERs where *vvv* would be a unique prefix value.

- *vv*0000-*vv*9999

  Defines a range of 10000 VOLSERs where *vv* would be a unique prefix value.

- *v*00000-*v*99999

  Defines a range of 100000 VOLSERs where *v* would be a unique prefix value.

Section: <VolumePoolDefinitions>

Feature: Volume Pooling

VolumeSize=

Overrides the virtual tape cartridge capacity of new scratch volumes. If you do not specify a value, the virtual tape cartridge capacity is the size that was specified during CA Vtape installation.

Valid Values: 400M|800M|2G|4G|8G|16G

Section: <VolumePoolDefinitions>

Feature: Volume Pooling

OSSCMDS Parmlib Member

The OSSCMDS Parmlib member contains operator console commands that CA Vtape processes after successful initialization. Any MVS, JES, and CA Vtape console command can be specified in this member. If you do not require commands at startup, code $NULL as the value for the StartupCommands attribute. The attribute is in the Parmlib Directory Section of OSPARMS Parmlib member.

To define the desired startup console commands, use the CMD or COMMAND attributes in the Startup Commands Section of the OSSCMDS member. For example:

```
<StartupCommands>
    CMD='D U,TAPE,ONLINE'
    CMD='&SVTS DISPLAY PARMLIB,SHORT,HARDCOPY'

    CMD='&SVTS DISPLAY STATUS'
    COMMAND='D A,L'
    COMMAND='&SVTS D A'
```

Note: The subsystem-ID symbolic &SVTS in the example ensures that the initializing Subsystem processes the SVTS command. If ‘SVT2 D A’ is coded, the command processes only when a Subsystem with a subsystem-ID of SVT2 is initializes.

OSPCMDS Parmlib Member

The OSPCMDS Parmlib member contains operator console commands that CA Vtape processes before starting the actual shutdown process. Any MVS, JES, and CA Vtape console command can be specified in this member. If you do not require commands at shutdown, code $NULL as the value for the ShutdownCommands attribute. The attribute is in the Parmlib Directory Section of OSPARMS Parmlib member.

To define the desired shutdown console commands, use the CMD or COMMAND attributes under the Shutdown Commands Section, as shown in the following example:

```
<ShutdownCommands>
     CMD='D U,TAPE,ONLINE'
     CMD='&SVTS DISPLAY PARMLIB,M,H'
     CMD='&SVTS DISPLAY STATUS'
     COMMAND='D A,L'
     COMMAND='&SVTS D A'
```

Note: The subsystem-ID symbolic &SVTS in the example ensures that the Subsystem that shuts down processes the SVTS command. If you **code** 'SVT2 D A', only when a Subsystem with a subsystem-ID of SVT2 is shut down is the command that is processed.

Chapter 12: Product Verification

This chapter provides the information necessary to customize the CA Vtape JCL procedure, start CA Vtape, and verify that it is in working order.

This section contains the following topics:

Rename SVTS and SVTSAS PROCs

In an environment where multiple CA Vtape Subsystems are active on the same LPAR, multiple PROCs with unique names can make it easier for the operations and technical support personnel to monitor and manage CA Vtape. To support this environment, you can rename the SVTSCE PROC.

If you want to rename either of these PROCS, then observe the following cautions:

- You cannot rename the PROC to a CA Vtape Subsystem-ID (SVT*n* where *n* = 1-8). This can cause the tasks to start under the MASTER subsystem instead of JES, resulting in JCL errors or other problems.
- If you rename SVTSAS PROC, set the SubaddressSpaceName attribute to &JOBNAME. The attribute is in the Startup Options Section of the OSPARMS Parmlib member.

If you rename or create multiple PROC sets, a pattern (SVTSCE1, SVTSCE2, SVTSCE3, and so on) allows for identification of the Subsystem and its sub address spaces. In these PROCs, the SVTS parameter is set to the corresponding subsystem-ID (SVTS=SVT1, SVTS=SVT2, and so on).

The last digit of the subsystem-ID in the PROC name reinforces the need to use the proper console command prefix or SVTS parm value when issuing commands or running reports to investige a particular subsystem. When investigating the SVTSCE2 tasks, issue console commands with a command prefix of SVT2 and use a parm of SVTS=SVT2 for utility jobs.

When starting or stopping the renamed PROCs, S/P SVTSCE*n*, where *n* is the last digit of the appropriate subsystem-ID would suffice. When restarting a Virtual Tape Controller address space, the command prefix would control the restart. SVT1 R CU=1 would restart SVTSCE1.SVT1V1, the Virtual Control Unit address space one for the SVT1 Subsystem.

Continuing with the example, the sub address spaces uses the same PROC as the main address space. If the PROC name is SVTSCE1 and two Virtual Control Unit address spaces that the main task starts, you see three started tasks executing on your system with job names of SVTSCE1. The main task has a step name of SVTS or SVT1. The controllers have step names of SVT1V1 and SVT1V2.

If you prefer that the sub address spaces use a different PROC name, you can copy the SVTSCE1 PROC to a new member (SVTSCE1A, for example). You then set the SubaddressSpaceName attribute in the Startup Options Section of the OSPARMS Parmlib member to a value of SVTSCE1A. When SVTSCE1 is started after the change, the main task is still SVTSCE1 with a step name of SVTS or SVT1. It starts the sub address spaces with the SVTSCE1A PROC so that the two sub address spaces have job names of SVTSCE1A with step names of SVT1V1 and SVT1V2.

Cutomize the PROCs

Follow these steps:

1. Copy the SVTSCE members from the HLQ.CCUUPROC library to the appropriate system PROCLIB.
2. Follow the comments in the members to customize them.

   At your site, if allocations with UNIT=SYSALLDA are not allowed in IGGPRE00 or Dynamic Allocation exit IEFDB401, you may need to add the following DD statement to the SVTS PROC:

   ```
   //SYSIN@  DD UNIT=your.DASD.esoteric,SPACE=(TRK,(1,1))
   ```
3. Update the SYSID=&SYSNAME parameter in the SVTS PROC to a unique value if all of the following are true:

   - CA Vtape is running on multiple LPARs that share the Global VCAT.
   - The operating system names of those LPARs are greater than six characters.
   - The first six characters of those operating system names are not unique.

   When a Subsystem is started, the operating system name is combined with a comma and the last digit of the subsystem name to create an internal system ID that is used to enqueue or reserve the Global VCAT and Virtual Volumes. If the system name is greater than six characters, it is truncated to six characters.

   For example, when Subsystem SVT1 is started on an LPAR named PRODSYSA, the internal system ID is "PRODSY,1". If Subsystem SVT1 is then started on PRODSYSB, its internal system ID is also "PRODSY,1". To prevent resource conflicts and corruption of the Global VCAT, each Subsystem checks for an enqueue with a QNAME of SVTSX and an RNAME of its internal system ID when started. If the enqueue is not detected, the Subsystem issues the enqueue to reserve its internal system ID value. If the Subsystem detects that its internal system ID value is already in use, the Subsystem terminates after issuing an SVT*n*I0151E error message.

To avoid this problem, update the SYSID=&SYSNAME parameter in the SVTS PROC to a unique value that is fewer than six characters. For example, on LPAR PRODSYSA, update the SVTS PROC to use SYSID=SYSA; on LPAR PRODSYSB, update the SVTS PROC to use SYSID=SYSB.

4. If you are going to execute the sub addresses spaces with a different job name, copy the SVTSCE PROC to a new member in the same PROCLIB after you have customized it. Locate the SubaddressSpaceName attribute in the Startup Options Section of the OSPARMS Parmlib member. Verify that the attribute is set to the new member name.

Start CA Vtape Subsystem

To start CA Vtape on the system console, enter S *procname*, where *procname* is SVTSCE or the renamed value that you chose.

Many messages are issued at startup to document whether key features are active or inactive. More messages are issued to document:

- Actions taken.
- If critical internal components are found.
- Status of the main task startup and its daughter tasks.

When the SVT1IRB00I Timer Manager Ready message is issued, the startup of all sub address spaces for a CA Vtape Subsystem is completed.

Verify Virtual Devices are Operational

Execute the SVT1 Display Active console command to display all the online and operational Virtual Devices.

Note: For a complete description of the command and sample output, see the chapter “Console Commands” in the *Administration Guide*.

If devices are missing, verify the output for the SVTSCE task that started for error messages. If there are no error messages, verify the OSDRIVE Parmlib member to ensure that the correct device range was properly defined.

Note: Modifications to the OSDRIVE member can be verified with the SVTPARMS syntax check JCL or by stopping and starting CA Vtape.

Verify CA Vtape is Operational

Multiple CA Vtape console commands are referenced in this section to display various installation-related items. These commands are documented in mixed case where the uppercase characters are those characters that must be entered for the Subsystem to identify the command.

Note: For complete descriptions of the commands and sample output, see the chapter "Console Commands" in the *Administration Guide*. If you detect problems during the verification steps, see the chapter "Troubleshooting" in the *Administration Guide*.

Follow these steps:

1. Issue the SVT1 Display Status console command to display the status of the Subsystem or started task. Check the following items:
   - Version is correct.
   - DSN prefix is correct.
   - Virtual Volume Size is correct.
2. Issue the SVT1 Display Parmlib console command. Check the following items:
   - ParmlibDirectory, SystemOptions, GroupDefinitions, VirtualDeviceList, and DatasetFilters. Are they set to the correct Parmlib member names?
   - Is an include filter coded?
3. Issue the SVT1 Display POOLS console command to display the defined pools and VOLSER ranges.
4. Submit a job using a data set name or data class that matches your test filter and use a UNIT parameter value that contains the Virtual Devices.

   The mount request should be intercepted and written to a Virtual Volume.
5. Submit a job to read the Virtual Volume and verify its content.
6. Display the Virtual Volume in the CA Vtape ISPF Interface.

   Edit HLQ.CCUUEXEC(SVTSMON) and complete the customization comments. Execute the customized SVTSMON member to start the CA Vtape ISPF Interface.

   Select option 3 to view the virtual VOLSER ranges. Select the range containing the Virtual Volume written. Select the actual Virtual Volume and review the data set name, DCB statistics, and other information. Compare this information with the information contained in the Tape Management System.

   In the top center of the display is a field that is named Flags. The field is 4 bytes and displays in hexadecimal format. The bits that make up this field document the status of the Virtual Volume. How to interpret this field and much more about the ISPF Interface is documented in the chapter "ISPF Interface" in the *Administration Guide*.

Product verification is complete. You are able to intercept a tape mount, and read/write to a Virtual Volume.

You were introduced to the ISPF Interface and various console commands. The SVT1 HELP console command lists the syntax of all CA Vtape console commands. For detailed command information and examples, see the *Administration Guide*.

If you are performing a trial installation, you have now installed and configured CA Vtape successfully to perform the trial. The following chapters of this guide provide jobs that you run for a production implementation. The chapters also provide topics concerning changes to the basic recommended configuration.

Chapter 13: Operational Considerations

This chapter describes aspects of the product related to day-to-day operation.

This section contains the following topics:

Control Data Set Backup

The CA Vtape BSDS is a critical system file. To ensure recoverability after a hardware failure, maintain a backup of the BSDS.

Back up the BSDS after batch processing completes. Then back up the Tape Management Database also.

A synchronized set of these backup data sets is not required. You can take the backups within a few minutes of each other.

Make the backups close together so that the current Virtual Volume information in the BSDS matches the Tape Management Database information. If the Virtual Volume information is too far out-of-sync, CA Vtape may not be able to access Virtual Volume files without manual intervention. Your disaster recovery efforts are then much more complicated.

Send these backups off-site for disaster recovery. Do not write the backups to CA Vtape Virtual Volumes.

Back Up the BSDS1

The ability to recover the Global VCAT from the BSDS is an important feature in CA Vtape.

Note: For more information, see the chapter “Recovering CA Vtape” in the *Administration Guide*.

We recommend that you maintain regular backups of the BSDS. In the unlikely event that both the BSDS and the Global VCAT are accidentally deleted or destroyed, you can restore the BSDS from a backup tape and then use it to recover the Global VCAT. You should back up daily to media that does not require that CA Vtape be active to read it.

Note: Because the Global VCAT can be recovered from the BSDS, a backup of both data sets is not required.

Back up the BSDS at about the same time as your tape management database. Taking backups of these data sets close together ensures that Virtual Volume information that is stored in the BSDS matches information that is stored in the tape management database. A synchronized set of backups is not required. If these backups are too far out-of-sync, it can complicate your disaster recovery efforts. CA Vtape can require manual intervention to access Virtual Volume files.

Back up the BSDS using CA Disk, DFSMSdss, FDR, or any other backup product that implements concurrent or snapshot copy. Concurrent or snapshot copy lets you create backups without having to stop CA Vtape or suspend tape processing.

Maintain backups on DASD or on physical tapes that provide access to the backup data set even when CA Vtape is inoperative. Using RAID devices for a backup to DASD or duplexing a backup to tape ensures that a subsequent media failure does not prevent the restoration of the most current Virtual Volume information.

The following is a sample of a CA Disk backup job:

```
//JOBNAME   JOB …
//BACKUP    EXEC DMS,MI=002
//SYSIN     DD *
  FIND      DSN=HLQ.BSDS1
  BACKUPCC  RETPD=14
```

The following is a sample of a DFSMSdss backup job:

```
//JOBNAME       JOB …
//BACKUP        EXEC PGM=ADRDSSU,REGION=4M
//SYSPRINT      DD SYSOUT=*
//INPUT         DD DISP=SHR,DSN=HLQ.BSDS1
//OUTPUT        DD DISP=(,CATLG),DSN=HLQ.BSDS.BACKUP(+1),
//LABEL=(1,SL),UNIT=REAL3490
//SYSIN         DD *
    DUMP DATASET(INCLUDE(HLQ.BSDS1)) -
    PHYSINDDNAME(INPUT) -
    OUTDDNAME(OUTPUT) -
    CANCELERROR OPTIMIZE(4)  CONCURRENT WAIT(20,20) -
    TOL(ENQF)
//
```

Recover the BSDS

If the Global VCAT is still available, use IDCAMS Repro to copy the Global VCAT into a newly defined BSDS.

If both the BSDS and the Global VCAT are lost but the network storage is intact, restore the BSDS from the latest backup and recover the Global VCAT as explained in the topic Recovering the Global VCAT.

To perform the restore, shut down all CA Vtape Subsystems participating in the CA Vtape Complex sharing the BSDS that you want to restore.

The following is a sample of a CA Disk restore job:

```
//JOBNAME  JOB …
//RESTORE  EXEC RESTORE
//SYSIN    DD *
  RESTORE  DSN=HLQ.BSDS1
//
```

The following is a sample of a DFSMSdss restore job:

```
//JOBNAME       JOB …
//RESTORE       EXEC PGM=ADRDSSU,REGION=4M
//SYSPRINT      DD SYSOUT=*
//INPUT         DD DISP=SHR,DSN=HLQ.BSDS.BACKUP(+0)
//OUTPUT        DD DISP=SHR,UNIT=3390,VOL=SER=XXXXXX
//SYSIN         DD *
  RESTORE DATASET(INCLUDE(HLQ.BSDS1)) -
    PHYSINDDNAME(INPUT) -
    OUTDDNAME(OUTPUT) -
    REPLACE
//
```

Recover the Global VCAT

If both the BSDS1 and the Global VCAT are lost but the network storage is intact, restore the BSDS from the latest backup and recover the Global VCAT.

Note: For specific steps to recover the Global VCAT, see the chapter “Recovering CA Vtape” in the *Administration Guide*.

To perform the recovery, shut down all CA Vtape Subsystems participating in the same CA Vtape Complex sharing the BSDS1 and Global VCAT that you want to recover.

Recover the Local VCAT

Because the Local VCAT contains transient data and is also a repository for Parmlib information, a backup is not required. If a Local VCAT is lost, you can customize and execute the HLQ.CCUUJCL(DEFVCAT) member to create a Local VCAT.

The CA Vtape Subsystem using this Local VCAT must be shut down during the recovery. Other CA Vtape Subsystems running on the same or different Logical Partitions can remain active.

After successfully defining and initializing the new Local VCAT, start the affected CA Vtape Subsystem.

Volume Contention

The operating system tries to serialize the use of tape volumes during allocation. You can control the way the operating system reacts to volume contention by customizing the VOLUME_ENQ attribute of SYS1.PARMLIB(ALLOC*xx*). The possible settings are CANCEL, WAIT, and WTOR. CA recommends specifying WAIT or WTOR and does not recommend specifying CANCEL.

- When set to CANCEL: the job that needs an unavailable volume will be cancelled.
- When set to WAIT: the job that needs an unavailable volume waits until the volume becomes available.
- When set to WTOR: the system issues message IEF235D when the contention is detected and allows an operator to end the wait. As soon as the volume becomes available the job continues and requires no operator action.

For a complete explanation of the VOLUME_ENQ attribute, see the IBM *z/OS MVS Initialization and Tuning Reference* manual. Volume ENQ Installation Exit (IEF_VOLUME_ENQ) can override VOLUME_ENQ behavior, as discussed in the IBM *z/OS MVS Installation Exits* manual.

You can configure CA Vtape to allow CA Disk data sets written to Virtual Volumes to be read by multiple CA Disk jobs. This feature is not supported on a JES3 system. This means a CA Disk Virtual Volume can be mounted for READ on multiple Virtual Devices at the same time.

CA Vtape interacts with the operating system to minimize the amount of time a CA Disk Virtual Volume is unavailable. Most tapes are unavailable until dismounted. A CA Disk Virtual Volume is only unavailable for the time needed to complete the allocation. Volume contention is less likely to occur except when jobs are submitted at the same time. If message IEF235D appears, it should last only a few seconds before the jobs proceed without operator intervention. CA Disk auto-restores (DMSAR) provide their own serialization and message IEF235D does not occur.

More information:

Concurrent Read Access to Virtual Volumes Created by CA Disk.

Preserving External Logger Data

The CA Vtape logging facility writes log events in to a dataspace. The logged data is small and is accumulated into 24576-byte logical blocks. When a logical block becomes full and external logging is active, the block is written to the IBM log stream that was defined during system setup.

To ensure that the data written out to the log stream is accessible for a longer period, we recommend that you archive the log stream data to Virtual Volumes.

You can find sample JCL for this task in HLQ.CCUUJCL(GENLOGR).

Saving the logged events for a longer period lets you use the data to look for trends, create historical reports, and diagnose problems.

More information

System Setup

Multiple CA Vtape Subsystems on the Same Logical Partition

CA Vtape can support up to eight concurrent subsystems for each Logical Partition. Subsystems are named SVT1, SVT2,…SVT8. Subsystems can be part of the same CA Vtape Complex. They can share the Global VCAT, BSDS, and network storage. Subsystems can also be independent with their own control data sets and network storage.

As delivered, the SVTSCE JCL procedure defaults to starting SVT1. To start a Subsystem with a different subsystem name, change the SVTS parameter in the SVTSCE JCL produced to SVT2 or SVT3, and so on. You do not have to have Subsystem SVT1 running before you can start Subsystem SVT2 or SVT3.

Running more than one Subsystem requires different LOCAL VCATs and Virtual Drives.

The SVTS= parameter on the EXEC statement of the SVTSSE JCL procedure in the HLQ.CCUUPROC library determines which Subsystem is started. The SVTS= parameter on the EXEC statement of the RECYCLE and SVTSUTIL jobs determines which Subsystem the job communicates with.

If the subsystems are part of the same CA Vtape Complex, you can achieve the same results running SVTSUTIL batch commands that point to SVT1 or any subsystem. If a Subsystem must be active for the utility to run, point to one of the active Subsystems.

Commands and messages are prefixed with the Subsystem Name by default. If you type SVTS for a command prefix, it defaults to SVT1. SVT1 executes and responds. To communicate with SVT2, use it as the command prefix. The CA Vtape Subsystem processing the command echoes the command using the appropriate prefix.

Command and message examples

```
SVTS D P
SVT1X0100I Command Complete

SVT8 D U
SVT8X0100I Command Complete
```

Copy CA Vtape Data Sets while in Use

You can copy the CA Vtape data sets while they are in use with the Transparent Data Migration Facility (TDMF) software. We recommend that you back up the BSDS before copying the Global VCAT or BSDS data sets. Perform any copy operations during a period of low CA Vtape mount activity.

Chapter 14: Customize Virtual Device Engine

This section contains the following topics:

Virtual Device Channel Paths

When the Virtual Devices are defined in the I/O Definition File (IODF), channel paths (CHPIDs) are not assigned to the devices. Instead CHPIDs are automatically assigned to the devices by CA Vtape during the first startup after an IPL. The CHPID assignment is critical. The operating system architecture allows an I/O device to be varied online only if the I/O device has a CHPID.

The CHPIDs chosen for automatic assignment to the Virtual Devices are the CHPIDs associated with the DASD volume the BSDS is allocated on. All CHPIDs associated with the BSDS DASD volume are assigned to each Virtual Device. A maximum of eight CHPIDs per Virtual Device are supported.

You can override Automatic CHPID assignment by coding either:

- (Preferred) The ChpidDeviceList attribute in the Virtual Control Unit section of Parmlib
- A CHPID DD in the SVTSCE PROC.

Code CHPID DD in the following ways:

- //CHPID DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=*vvvvvv*, where *vvvvvv* is a DASD volume whose CHPIDs are used to override Automatic CHPID assignment.
- //CHPID DD DISP=SHR,DSN=*your.data.set*, where *your.data.set* has been allocated on the DASD volume whose CHPIDs are used to override Automatic CHPID assignment.
- //CHPID DD DISP=SHR,UNIT=(/*uuuu*,,DEFER),VOL=SER=VTAPES, where /*uuuu* is the four-digit unit address (preceded by a forward slash) of the tape drive whose CHPIDs are used to override Automatic CHPID assignment.

Note: When trying to use a tape CHPID, the unit specified on the CHPID DD must be online to the system CA Vtape is started on.

The ChpidDeviceList attribute can be coded with a list of online devices with CHPIDs that should be used as the Virtual Device CHPIDs.

After the Virtual Device CHPIDs are assigned, automatically or manually, they are remembered across CA Vtape warm starts. Adding or removing the CHPID DD or the ChpidDeviceList attribute has no effect until the next cold start. Cold starts only occur after the system is IPLed.

If the Virtual Device CHPIDs must be changed, stop all CA Vtape virtual tape activity, stop CA Vtape, change the VirtualDeviceList attribute in the Parmlib Directory Section of the OSPARMS member to point to the VTDRALT member, and start CA Vtape. The VTDRALT member contains an unused set of device addresses defined for use by CA Vtape during initial customization. Because these devices have never had CHPIDs assigned to them, CA Vtape can assign CHPIDs to these devices and vary the devices online without violating the operating system architecture and without the need for an IPL.

Overriding the default CHPID choice may be required due to a channel problem, a DASD problem, or a microcode upgrade to DASD hardware that causes the hardware to take its CHPIDs offline during the upgrade process. When overriding the default CHPID choice, the best CHPID to pick is the one that has the least likelihood of being taken offline. Since the CHPID is only assigned to satisfy the operating system architecture and is not used to perform any I/O's, the CHPID chosen does not experience an increase in I/O activity that could degrade the performance of the other devices that actually use the CHPID for I/O.

If you modify the CHPID DD to point to a different device and you restart CA Vtape without an IPL, and continue using the same Virtual Devices, the devices may fail to vary online. The system always adds new CHPIDs to the already-defined CHPIDs and as a consequence the maximum number of eight CHPIDs may be easily exceeded. In this situation the only solution is to use the alternate set of Virtual Devices (member VTDRALT) or IPL the system.

An alternative solution would be to set up and run two different CA Vtape Subsystems sharing the same Global VCAT and BSDS1. The SVTSCE PROC would be copied to create an SVTSCE1 and an SVTSCE2 PROC. Each PROC would have a CHPID DD coded. The CHPID DD's would reference DASD volumes in different hardware subsystems so that all the Virtual Devices are not using the same CHPIDs.

This would allow you to vary offline the Virtual Devices for SVTS1 which are using the CHPID's associated with one DASD hardware subsystem. A microcode upgrade which takes the CHPID's offline could then be safely performed on that DASD hardware subsystem. Once the upgrade was complete, the SVTSCE1 Virtual Devices could be varied online. The SVTSCE2 Virtual Devices would then be varied offline and the other DASD hardware subsystem upgrade would be performed. When that upgrade is complete, the SVTSCE2 Virtual Devices could be varied online.

We recommend specifying the ChpidDeviceList attribute to assign non-DASD CHPIDS to Virtual Devices. This avoids having to set up multiple subsystems and prevents CA Vtape Virtual Devices from being involved in any DASD channel recovery events.

Note: DASD subsystems containing paging or XCF data sets should be avoided. The best candidates are CTC (Channel-To-Channel) and OSA (Open Systems Adapter) CHPIDs.

A best practices approach is to spread the Virtual Devices across multiple Virtual Control Unit address spaces and then spread the CHPID utilization across multiple non-DASD devices using the ChpidDeviceList attribute.

CHPID Failover

If one of the CHPIDs that the Virtual Devices are using is taken offline, CA Vtape interacts with the channel path recovery feature of the operating system to switch dynamically to the next available CHPID in the assigned list. As with all devices, when the last assigned CHPID goes offline, the affected Virtual Devices are boxed or fenced. If the Virtual Devices are in use, take the following steps:

1. Cancel the affected jobs.
2. Bring back online at least one of the assigned CHPIDs.
3. Stop and start CA Vtape.
4. Resubmit the affected jobs.

The CHPID list is one directional. When the first CHPID in the list is taken offline, the Virtual Devices are failed over to the next CHPID in the list. When the second CHPID in the list is taken offline, the Virtual Devices are failed over to the third CHPID in the list. Bringing the first CHPID in the list back online does not matter. Failover cannot go back to a CHPID previously used. Failover also cannot wrap back to the first CHPID in the list when the last CHPID in the list is taken offline. After the CHPID failover occurs, the failed CHPID can only be reused if CA Vtape is stopped and started.

Virtual Volume Compression

Virtual Volume compression lets you compress data that is written to Virtual Volumes. Compressed volumes occupy less disk space. Compressed data requires fewer I/Os to read or write at the expense of more CPU overhead to perform the actual data compression.

Virtual Volume Compression is activated and controlled through attributes that are defined in the Dynamic Options Section of the OSPARMS member of Parmlib. The following Parmlib attributes control how and when compression is activated:

- HardwareCompressionOption=
- MinimumCompressionRate=
- MaximumCompressionCPU=
- HardwareCompressionMethod=

How Virtual Volume Compression Works

When compression is active, CA Vtape always compresses the first 5 MBs of every 50 MBs of data that is written to the Virtual Volume. CA Vtape always compresses the first 5 MB of data to determine the best compression method for the remaining 45 MB of data. If the compression rate falls below the MinimumCompressionRate setting, then CA Vtape discontinues compression until the next 50 MB segment is written.

When you set the MinimumCompressionRate to a low value, CA Vtape compresses a larger percentage of the Virtual Volumes and more CPU is used for compression. When the MinimumCompressionRate is set to a high value, CA Vtape compresses a smaller percentage of the Virtual Volumes and uses less CPU for compression.

Compression reduces I/O and storage space at the cost of more CPU. The compression routines in CA Vtape rely heavily on the use of certain hardware instructions to perform compression. The CPU expense for using these instructions varies based on the CPU hardware in your environment. In some instances, the additional CPU expense might not justify the use of compression.

MaximumCompressionCPU lets you control this additional CPU expense by controlling the percentage of data compression is performed on:

- **100**: CA Vtape compresses all data that is written to a Virtual Volume. 100 percent produces the maximum compression at the highest CPU cost.
- **50**: CA Vtape compresses 50 percent of the data that is written to a Virtual Volume. On average this results in approximately 50 percent less compression at 50 percent less CPU cost.
- **0**: CA Vtape does not compress data except for the first 5 MBs of each 50 MB of data written.

If a zIIP Engine is installed (and available) and the zIIPExploitation and PercentRunOnzIIP Parmlib attributes are set appropriately, the compression work is reduced, lowering the CPU compression cost.

Virtual Volume CA Tape Encryption Interface

Any tape, either virtual or physical, can be encrypted with CA Tape Encryption. If encryption is activated system-wide for all tapes, CA Vtape Virtual Volumes are encrypted when written by the applications.

The AllowBTEencryptionForVVE attribute in the Dynamic Options Section of the CA Vtape OSPARMS Parmlib member, lets you control whether Virtual Volumes are encrypted by CA Tape Encryption.

For a detailed description of this attribute, see the chapter “The Parameter Library (PARMLIB).”

Concurrent Read Access to Virtual Volumes Created by CA Disk

You can mount most tape media on only one device at a time. This forces applications like CA Disk to serialize tape access when accessing multiple files that reside on the same tape volume (CA Disk ARCHVOL). For instance, multiple DMSAR subtasks or batch Restore jobs that are directed to restore different user DASD files residing on the same tape volume, must wait and access the tape volume one at a time.

If you are running on a JES2 system, you can enable concurrent read access to CA Disk ARCHVOLS that reside on Virtual Volumes. Set AllowConcurrentVVEReadAccess to CADISK in the Dynamic Options Section of the OSPARMS Parmlib member.

Note: CA Disk ARCHVOL Restore processing exploits AllowConcurrentVVERead Access. Other CA Disk utilities exploit AllowConcurrentVVERead. If simultaneous allocations for the same Virtual Volume occur, the utilities can require your automation software to respond WAIT to message IEF690I.

Concurrent Read Access lets you write CA Disk ARCHVOLS to CA Vtape Virtual Volumes. You benefit from tape utilization efficiencies and operational performance improvements during disaster recovery and day-to-day storage management processes.

Chapter 15: Understanding Performance

Because CA Vtape is a software virtual tape solution, every function the product performs requires CPU and memory. If either of these resources become constrained, it can elongate mount times, virtual tape processing times, and job run-times.

This section contains the following topics:

Response Time

The CA Vtape response time is directly related to available MIPS, Linux, and the DASD response times.

If the CPU is constrained, CA Vtape runs slower like the other software and jobs executing on the LPAR. Jobs attempting to mount Virtual Volumes experience elongated mount and run-times. You can help in this situation by:

- Increase the available MIPS.
- Move workloads that do not use CA Vtape to other LPARs.
- Install a zIIP engine.

If the DASD used for CA Vtape has slow response times, then jobs attempting to read and write Virtual Volumes experience elongated run-times. You can address slow DASD response times in a number of ways. The following are some of the most effective ways:

- Use OSA cards.
- Add memory or DASD storage for the DASD appliance.
- Provide DASD appliances through extra mount points.

Exploit zIIP Specialty Processors

CA Vtape is designed to exploit IBM zIIP specialty processors when they are installed and enabled on the system where CA Vtape is active and the Parmlib attributes are configured correctly. By default, CA Vtape will not attempt to make work eligible to run on the zIIP. You must activate zIIP processing using the zIIPExploitation= and PercentRunOnZIIP= attributes in Parmlib.

The Virtual Device Engine for CA Vtape is designed to run in Service Request Block (SRB) mode, a prerequisite of zIIP processing. When the zIIP feature is activated, CA Vtape will create the environment necessary to schedule work on the zIIP. All Virtual Device Engine processing, including compression and decompression processing, runs in SRB mode; therefore the most processor intensive functions of CA Vtape can be made eligible to run on the zIIP.

When the zIIP feature is activated, CA Vtape creates the environment necessary to enable all or a percentage of its Virtual Device Engine work for zIIP processing.

For example, when you specify zIIPExploitation=Y to active zIIP support and PercentRunOnZIIP=20, every fifth Virtual Device Engine SRB will be enabled for zIIP processing.

Note: IBM documents that work enabled to run on a zIIP processor may not be processed on the zIIP for a number of reasons. For more information about the zIIP, see the appropriate IBM documentation.

Monitor zIIP Utilization for a Service Class

Usage of the zIIP specialty processor should be planned and monitored through the IBM Resource Management Facility (RMF) Service Class report (WLMGL). This report is based on data taken from the following SMF Record Types:

- SMF Record Type 72 (x48) RMF Workload Activity and Storage Data
- SMF Record Type 79 (x4F) RMF Monitor II Activity

The Service Class report provides information to your capacity planner to determine how much zIIP capacity is needed. zIIP utilization will be reported for the SVTSCE.SVT*n*V*n* address spaces only.

When viewing the RMF Service Class report fields; CP, IIP, and IIPCP in the *APPL %* column are most important:

CP

Represents the percentage of work run on a general CP.

IIP

Represents the percentage of work run on a zIIP processor.

IIPCP

Represents the percentage of work eligible for the zIIP processor that WLM redirected to the general processor.

If the zIIP Exploitation feature is activated, you should see a percentage of work in the IIP field. When zIIP processors are online and CA Vtape is exploiting them you may see some reported in the IIPCP field. It represents the percentage of work that was enabled to run on the zIIP engine, but was redirected by WLM to the general CP.

You can perform capacity planning activities before you have zIIP processors installed or online by running with zIIPExploitation=Y and PercentRunOnZIIP=100. CA Vtape will enable its work for the zIIP engine and RMF will report that work in the IIPCP column. If IIPCP is >= 100% that means more than one zIIP processor would be needed for the work that was zIIP enabled. If IIPCP is >= 100% that means more than one zIIP processor would be needed.