CA Vtape™ Virtual Tape System

Configuration Guide

Release 12.6.00, Third Edition

CA Technologies Product References

The CA Vtape™ Virtual Tape System guides refer to the following CA products and components:

- CA 1® Tape Management (CA 1)
- CA Allocate™ DASD Space and Placement (CA Allocate)
- CA Compress™ Data Compression (CA Compress)
- CA Earl® (CA Earl)
- CA Graphical Management Interface (CA GMI)
- CA Chorus Software Manager™ (CA CSM)
- CA MIM™ Resource Sharing (CA MIM)
- CA Sort® (CA Sort)
- CA Tape Encryption
- CA TLMS® Tape Management (CA TLMS)
- CA Vantage™ Storage Resource Manager (CA Vantage)
- CA Vtape™ Virtual Tape System (CA Vtape)
- CA Vtape™ Virtual Tape System Peer-To-Peer Option (CA Vtape P2P)

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: Overview 11

Audience 11
Configuration Process 11
How to Use this Guide 12

Chapter 2: System Setup 13

Multisystem Planning 13
Multisystem Requirements 14
Reserves and Enqueues 15
    SVTS Reserve 15
    SVTSX and SVTRCYCL Enqueues 17
JES3 Requirements 17
Define Virtual Devices Using IBM's Hardware Configuration and Definition (HCD) Dialogs 18
Work Load Manager (WLM) Considerations 19
APF Authorizations 19
DASD Allocation Control Products 19
Saving the Logger Data 19
    The IBM System Logger 20
    Log Stream General Considerations 21
    Defining the Log Stream 21
Tape Management System Considerations 23
    Sharing the Tape Management System Database 23
    Defining an Exclusive Tape Range for the Virtual Volumes 24
    Subpooling 24
    Support for Data Set Stacking 25
VTA08NON Job to Allocate Non-SMP/E Data Sets 25
Oracle Host Software Component (HSC) and Storage Management Component (SMC) 26
    Method One: POLICY Statements 26
    Method Two: HSC/SMC Exits 26
    Method Three: HSC/SMC Parameters 28
    Method Four: SMS and CA Allocate 29
Integration with CA OPS/MVS 29
    Enable CA OPS/MVS Event Notification 30
    CA Vtape System State Management (SSM) 30
    CA Vtape Health Check State Management 30

Chapter 3: Cache Management 31

Determining the Global VCAT and BSDS Size 31
Determining the Cache Size 32
Understanding the IBM Volume Mount Analyzer (VMA) 32
Running Product Modeling 32
Deciding on a Cache Management Strategy 34
Dynamic Cache Management 35
Setting Up Dynamic Cache Management in SMS....36
DASD Allocation Control Products 40
Dynamic Cache Virtual Volume Size....40
Static Cache Management 41
Static Cache Virtual Volume Size....41
Determining the Number of Static Cache DASD Volumes 42
Setting up Static Cache Management 43

Chapter 4: ISPF Customization Panels 45

Product Customization Panels 45
ISPF Customization Panel Primary Options 46
ISPF Customization Steps 47

Chapter 5: Control and Cache Data Sets 55

Control Data Sets....55
Define the Global VCAT and BSDS....56
Define the Local VCATs 56
Define the Cache Data Sets 57

Chapter 6: Tape Mount Intercept Filters 59

General Description 59
Tape Mount Redirection 59
Using SMS Data Class Constructs 59
Using Data Set Name Filtering 60
Using Both SMS Data Class Constructs and Data Set Name Filtering 60
Using Esoterics or Generics....61
Tape Mount Intercept Filters 62
VTFILTR Parmlib Member....62
Date Set Name Pattern Masking 63
Filter Processing 64

Chapter 7: Scratch Tape Synchronization 67

Tape Expiration 67
Scratch Tape Synchronization 67
CA 1 Tape Management System 68
CA 1 ROBSCR Option 68
CA 1 Scratch Synchronization Job 69
CA 1 DSNB Records 69
CA TLMS Tape Management 70
DFSMSrmm....70
Control-T (BMC) 70
AutoMedia (Zara) 70
Other Tape Management Systems....71
Scratch Pool Processing....71
VTPOOL Parmlib Member 71
SVTn ADD VVP Console Command....72
CA Vtape P2P Option Scratch Pool Processing 73

Chapter 8: Tape Management Systems 75

Stacked or Multiple-File Tape Support....75
CA 1 Tape Management System 75
CA TLMS Tape Management 76
DFSMSrmm 76
Control-T (BMC) 76
AutoMedia (Zara) 77
Tape Management System Interface 77
Virtual Volume Expiration Date and Automatic Subgroup Adjustment....77
Virtual Volume Stacking Location 78
Foreign Tape Processing....78

Chapter 9: Configuring the Parameter Library (Parmlib) 81

General Description 81
Parmlib Syntax....82
Parmlib and Symbolic Substitution 83
Parmlib Attribute Values....83
Symbolic Substitution for Parmlib Member Names....84
Initial Configuration of the Parmlib....85
Parmlib Syntax Verification 88
Automatic Command Execution....89

Chapter 10: The Parameter Library (PARMLIB) 91

Parmlib Attribute Feature and Location....91
Parmlib Attributes Related by Feature....96
VTPARMS Parmlib Member....101
PARMLIB DIRECTORY....101
STARTUP OPTIONS....104
DYNAMIC OPTIONS....116
VTDRIVE Parmlib Member....139
VTGROUP Parmlib Member....142
Group Definitions....142
Group Sections....143
VTP2POPT Parmlib Member....159
Peer To Peer Options Section....159
VTP2PRMT Parmlib Member....163
Peer To Peer Remotes Section....163
Remote Section....164
VTFILTR Parmlib Member....166
Data Set Filters Section....167
Include Data Class Section....169
Include Data Sets Section....170
Exclude Data Sets Section....170
VTPOOLS Parmlib Member....171
Volume Pool Definitions Section....171
VolserRange Section....172
VolumePools Section....173
VTSCMDS Parmlib Member....174
VTPCMDS Parmlib Member....175
VUSSMNTS Parmlib Member....175
Mount Point Directory Sections....175

Chapter 11: Product Verification 179

Renaming the SVTS and SVTSAS PROCs....179
Customizing the PROCs....180
Starting the CA Vtape Subsystem....181
Submitting the Static Cache LDSADDxx Jobs....182
Verifying the Virtual Devices Are Operational....182
Verifying CA Vtape is Operational....182

Chapter 12: Operational Considerations 187

Control Data Set Backup....187

Backing Up the BSDS 188
Recovering the BSDS 189
Recovering the Global VCAT 189
Recovering the Local VCAT 190
High-Speed Open (HSOPEN) Option 190
Volume Contention 191
Preserving External Logger Data 192
Multiple CA Vtape Subsystems on the Same Logical Partition 192
Copying CA Vtape Data Sets While They Are In Use 193

Chapter 13: Virtual Device Engine 195

Virtual Device Engine Implementation and Use 195
How Many Virtual Control Units? 196
Virtual Device Channel Paths 197
CHPID Failover 199
Virtual Volume Compression 200
How Virtual Volume Compression Works 200
Virtual Volume CA Tape Encryption Interface 201
Concurrent Read Access to Virtual Volumes Created by CA Disk 201

Chapter 14: The Backstore Engine 203

Starting, Stopping, and Restarting the Backstore Engine 203
Backstore Tape Capacity and Usage 204
Externalization Server Subgroup Queue Management 205
Default Group and Subgroup 206
Subgroup Automation 207
Overriding and Resetting Subgroup Automation 209
Configuring the Backstore Engine in Primary and Failover Roles 211
Backstore Externalization Throughput 212
Export Tapes 212
Releasing Tapes from Backstore Processing 213
Limiting the Number of Tape Drives used by the Backstore Engine 214
FullMaxdrivesEnforcement Set to No 214
FullMaxdrivesEnforcement Set to Yes 215
Cache Shortage 215
MAXDRIVES Setting 215
Automatic Recall Source Switching 216
Allocation Logic - CA MIM or IBM GRS Support 217
BypassOfflinePhysicalDevices=Y 218
BypassOfflinePhysicalDevices=N 218
Contention Over Backstore Tapes 219

USS Backstore 220
How the USS Backstore Works 221
Implementation Required Skills 222
USS File Systems 222
Security Considerations 223
Configure the USS Backstore File System 224
USS Backstore Advanced Configurations 227
NFS Performance 229
File System Reconfiguration 230

Chapter 15: CA Vtape P2P Option 233

Overview and Requirements 233
How P2P Works 234
Scratch Mount Requested by a Remote Virtual Tape Unit 234
P2P Security Considerations 235
Configure for CA Vtape P2P Subsystems 236
Set Up a Simple P2P Configuration to Test TCP/IP Connectivity 237
Set up a SVTS Subsystem in a Separate CA Vtape Complex 241
Test Remote Virtual Volumes Access 243
CA Vtape P2P Advanced Configurations 250
Scratch Processing for Remote Virtual Volumes 251

Chapter 16: Performance 253

Response Time 253
Exploitation of the zIIP Specialty Processor 254
How to Monitor zIIP Utilization for a Service Class 254
How to Monitor zIIP Utilization 255
CPU Performance Compared to Memory Utilization 256
STANDARD Mode 256
ISOLATION Mode 257

Index 259

Chapter 1: Overview

This guide describes how to configure CA Vtape.

This section contains the following topics:

Audience (see page 11)
Configuration Process (see page 11)
How to Use this Guide (see page 12)

Audience

This guide provides system programmers and storage administrators the information needed to configure CA Vtape.

Users of this guide should be experienced mainframe technicians. Knowledge of your mainframe tape system, tape related software, and security setup is essential for configuring the product successfully.

Configuration Process

You must complete the installation instructions provided in the *Installation Guide* before you can perform the configuration in this guide.

The overall configuration process is as follows:

1. Perform various critical tasks documented in the chapter "System Setup" to define CA Vtape to the operating system, the tape management system, and other third party software.
2. Review the chapter "Cache Management" and determine the type of cache to use.
3. Review the chapter “ISPF Customization Panels” and use the ISPF customization panels to create JCL to define the control data sets, to define the Static Cache data sets (only if Static Cache is to be used), and to create the SUTPARMS configuration member.
4. Review the chapter "Control and Cache Data Sets" and define the necessary data sets.
5. Review the chapter "Tape Mount Intercept Filters" to create a filter strategy.
6. Review the chapter "Scratch Tape Synchronization" and set up the required scratch synchronization job.

7. Review the chapter "Tape Management Systems" and how tape stacking and foreign tape processing is supported.
8. Review the chapter "Configuring the Parameter Library (PARMLIB)" and customize your parmlib.
9. Review the chapter "The Parameter Library (PARMLIB)" as needed to gain a more in-depth understanding of the parmlib members and their contents.
10. Follow the instructions in the chapter "Product Verification" to customize the CA Vtape procedures, start the product, and verify the product's operation.
11. Review and follow the instructions in the chapter "Operational Considerations" to set up utility JCL to back-up the control data sets, adjust the parmlib settings, and adjust other software to better handle your environment and needs.

The *Administration Guide* contains additional information which you may need when following instructions in this guide. Such as:

- An overview of the product, its features, and its components
- The CA Vtape console commands
- The CA Vtape utilities and reports

How to Use this Guide

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

- Variable text is entered in italics. This is most commonly used for dataset names and console commands.

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

  Commands are entered in uppercase and lower-case. The uppercase portion of the command is the abbreviated form of the command or the minimum number of characters that must be entered for the product to recognize the command. The lower-case portion is provided for clarity.**Note:** For other console command syntax conventions, see the chapter "Console Commands" in the *Administration Guide*.
- Features, functions, and components of [set to your product name] are capitalized. These include, for example: Virtual Volumes and Virtual Devices.

Chapter 2: System Setup

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

This section contains the following topics:

Multisystem Planning (see page 13)
Multisystem Requirements (see page 14)
Reserves and Enqueues (see page 15)
JES3 Requirements (see page 17)
Define Virtual Devices Using IBM's Hardware Configuration and Definition (HCD) Dialogs (see page 18)
Work Load Manager (WLM) Considerations (see page 19)
APF Authorizations (see page 19)
DASD Allocation Control Products (see page 19)
Saving the Logger Data (see page 19)
Tape Management System Considerations (see page 23)
VTA08NON Job to Allocate Non-SMP/E Data Sets (see page 25)
Oracle Host Software Component (HSC) and Storage Management Component (SMC) (see page 26)
Integration with CA OPS/MVS (see page 29)

Multisystem Planning

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

- The Global VCAT provides the mechanism by which the DASD buffer and the Virtual Volumes are shared between multiple CA Vtape Subsystems.
- A CA Vtape Complex is created by sharing a Global VCAT between multiple Subsystems running on one or more systems.
- More than one Complex can be created by creating and sharing different Global VCATs. This is typically only done to separate test and production data.
- For performance reasons, CA Vtape utilizes hardware reserves against the DASD volume on which the Global VCAT resides to serialize updates.
- The Global VCAT and Bootstrap data sets (BSDS) should never be allocated on the same DASD volumes as the dynamic cache data sets. The hardware reserve to update the Global VCAT will conflict with certain IDCAMS functions used to manipulate the dynamic cache data sets which could cause performance problems or a deadly embrace.

- 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 since they are being emulated by separate started tasks. Both devices can be online and in use at the same time on their respective systems.
- Virtual Volumes can be shared across multiple subsystems just like physical tapes and are serialized just like physical tapes.
- Concurrent read of a Virtual Volume is allowed if the reader is CA Disk. For instance, 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 will allow the auto-restores to access that Virtual Volume at the same time.
- Virtual Volumes may be Externalized by any active Externalization Server in a CA Vtape Complex and recalled by any active Recall Server in the Complex.
- The RECYCLE utility can be run on any system to consolidate fragmented Backstore Tapes.
- The CA Vtape parmlib supports symbolic substitution in its parmlib member names and parmlib attribute values. This support will allow you to use a single parmlib to support a 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:

- The Global VCAT and BSDS must be allocated on separate DASD volumes shared by all systems.
- If multiple CA Vtape Complexes will be defined, the Global VCAT and BSDS1 data sets pertaining to the different Complexes must be allocated on separate DASD volumes shared by all systems.
- The DASD buffer must be on shared DASD to allow Virtual Volumes to be read and written by all CA Vtape Subsystems in the same Complex.
- The shared DASD buffer must be configured to provide adequate performance for each CA Vtape Subsystem in the same Complex.
- The ICF catalog containing the DASD buffer and Externalized Virtual Volume entries must be on shared DASD and connected to all systems in the same Complex.
- Sysplex is supported, but not required. If available, its features can be exploited to reduce Recall times.
- Cross-system enqueue propagation is not required, but will be used, if available, by Externalization, Recall, and Recycle to resolve Backstore Tape contention.

- If a Tape Management System is installed, all Subsystems in the same CA Vtape Complex should share a common Tape Management System Database.
- If Subsystems on different systems, but in the same Complex, will be configured as Externalization or Recall Servers, the Backstore Tapes must be shared across those systems.

Reserves and Enqueues

In this section one reserve and two enqueues are discussed.

SVTS Reserve

CA Vtape keeps its control information in a dataspace and writes it out to the Global VCAT and BSDS1 data sets. To ensure the integrity of these updates, a hardware reserve is issued with QNAME=SVTS against the volume on which the Global VCAT resides.

The RNAME value used is determined by setting the GlobalReserve attribute located in the Dynamic Options Section of the VTPARMS 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 Subsystem 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 BSDS1 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, because by default hardware reserves are not automatically converted.
- For IBM GRS, you do not need any GRSRNL00 parmlib member updates, since 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)
  ```

SVTSX and SVTRCYCL Enqueues

Backstore Tapes containing Externalized Virtual Volumes can be enqueued by Recycle and the Backstore Engine Externalization and Recall functions, all running on different systems. These three functions can try to mount the same tape at the same time and experience the standard wait for an in-use tape volume. This delay is eliminated by always propagating the QNAME SVTSX and SVTRCYCL enqueues as SCOPE=SYSTEMS enqueues to all systems. This allows Recall to take a tape away from Recycle or Externalization. This ensures that an application job or task always has priority over CA Vtape secondary functions.

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

A primary and alternate set of Virtual Devices must be defined. The primary set will be used for virtual tape processing. The alternate set will be available in case of a permanent failure of the primary set that can only be cleared by an IPL.

To define the Virtual Devices to the operating system, follow the Define I/O Device Data procedure in the *IBM HCD User's Guide* with the following changes:

1. The devices will be defined as 3480s or 3490s because CA Vtape emulates only these tape device types.
2. The devices will not be defined with control units (CUs) or channel paths (CHPIDs) because they are software devices.
3. On the Define Device Parameters/Features screen, make sure that Autoswitch is No, Dynamic is Yes, Sharable is No, and Compact is Yes.
4. Add the newly defined Virtual Devices to new or existing esoterics for which virtual mounts are desired. We recommend that existing esoterics be updated to avoid JCL changes.
5. Follow the Build a Production IODF procedure in the *IBM HCD 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 HCD User's Guide*. Any change to the HCD software or the *HCD User's Guide* may invalidate this information.

The primary and alternate set of devices defined will be entered into the VTDRIVE and VTDRALT parmlib members. For more information, see the section Initial Configuration of the Parmlib (see page 85).

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

APF authorize 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.

DASD Allocation Control Products

If CA Vtape was installed with Dynamic Cache Management, the CA Vtape LDS allocations requested by the SVTS and SVTSAS started tasks should be exempted from third party DASD allocation control products. Products which intercept space abends or adjust allocation amounts based on DASD pool conditions are not needed since Space Constraint Relief is coded in the SMS data class defined for CA Vtape use. Also, CA Vtape requests specific primary and secondary space values based on the CachePrimary and CacheSecondary attributes coded in its parmlib and takes appropriate actions when these amounts cannot be honored. Adjusting these values with a third party product could result in CA Vtape taking inappropriate actions or an ABEND.

Saving the Logger Data

CA Vtape 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, always use the SVT*n* DUMP console command when capturing diagnostic information about CA Vtape.

CA Vtape treats this dataspace like a wrap-around file. When it 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. 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.

The IBM System Logger

To define a Log Stream, the IBM System Logger must be active. It is beyond the scope of this document to instruct you on setting up the IBM System Logger facility. 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 have been 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 CA Vtape 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*.

Defining the Log Stream

The External Logger requires the creation of 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 CA Vtape 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 System Considerations

If a Tape Management System is installed, the following must be considered at this point in the system setup:

- If CA Vtape will be run on multiple systems and the Virtual Volumes are to be shared across those systems, the Tape Management System Database should be shared between those systems.
- An exclusive tape VOLSER range must be defined in the Tape Management System for CA Vtape use.
- If the Virtual Volumes will be defined in subpools.
- How data set stacking is supported.

Note: Additional Tape Management System considerations are discussed in the chapters “Scratch Tape Synchronization" and "Tape Management Systems.”

Sharing the Tape Management System Database

Virtual Volumes can be shared between systems, just like physical tapes. Also, just like physical tapes, if the Tape Management System (TMS) Database is not shared, tape usage is not completely documented, tape security may not be flawless, 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 will not be documented in the Database. Since the system B Database is not controlling the usage of this tape, an unauthorized user may be able to read or write to this tape. If the TMS is setup to not allow the mounting of tapes that are not documented in the Database, a TMS override may need to be coded in the application JCL to mount the tape on system B.

If the CA Vtape Subsystems running on system A and system B are configured as Externalization and Recall Servers, the Backstore Tapes must be shared. If the TMS Database is not also shared and the TMS is setup to not allow undocumented or foreign tapes to be mounted, the Backstore Tapes will only be mountable on the system where the tapes are defined to the TMS. An override cannot be coded since CA Vtape always performs a dynamic allocation of the Backstore Tapes.

Defining an 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.

Subpooling

Subpooling is not required, but is supported by CA Vtape. If you have defined subpools in your Tape Management System for the workloads that will be moved to CA Vtape, those subpool definitions must be duplicated in the subpool definitions in the CA Vtape VTPOOLS parmlib member.

Note: Multiple pool definitions are not supported under JES3. Under JES3 only one volume pool can be defined.

Up to eight subpools can be defined. If the workloads that will be moved to CA Vtape currently use more than eight subpools, then you will need to consolidate your Tape Management System subpool definitions to reduce the number of subpools to eight or less. Consult your Tape Management System manuals to determine how to modify your subpool definitions.

If you have licensed the CA Vtape Peer-To-Peer Option, then you will need to reduce the number of subpools defined to seven or less. Subpool eight is reserved for the Peer-To-Peer Option.

Note: How to define the subpools is covered in the chapter "The Parameter Library (PARMLIB)."

Support for Data Set Stacking

The CA Vtape Backstore Engine stacks Virtual Volumes as data sets on the Backstore Tapes. Each Virtual Volume stacked is cataloged as an independent data set with a generated name and catalog managed retention. These catalog entries are collectively known as the Backstore Entries.

Each Backstore Entry will be tracked by the Tape Management System. This causes an increase in the number of records for data sets maintained in the Tape Management System Database. Review the actual space available in the database to determine if it will support the expected increase in the number of records.

Review the Tape Management System documentation and setup to ensure that tapes are returned only to scratch status when all the data sets stacked on them have expired. Without the proper setup, the Tape Management System may scratch a stacked tape when the first data set (file 1) on the tape expires. This could cause a *data loss* situation if files 2 through '*n*' have not expired. This applies to both virtual and physical tapes, because applications can write multiple data sets to a Virtual Volume, just like they do to a physical tape.

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:

- Code POLICY statements.
- Install the HSC/SMC exits.
- Update HSC/SMC parameters and allocation routines.
- Use SMS or CA Allocate to do unit replacement.

The recommended method is using POLICY statements.

Method One: POLICY Statements

The latest releases of the HSC/SMC software no longer support 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, consult the HSC/SMC manuals for the installed release of the software.

Method Two: HSC/SMC Exits

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

The SLSUX08 exit 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: HSC/SMC Parameters

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*, a Recycle of the HSC started task is required 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: SMS and CA Allocate

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

In this approach, a separate unit esoteric is coded for the CA Vtape Virtual Devices and is 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, SMS 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 will 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, SMS 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

The interface is enabled by setting the CA OPS/MVS parameter APIACTIVE to ON. This allows CA OPS/MVS to acknowledge and process the events generated by CA Vtape.

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.

CA Vtape 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: Cache Management

This section contains the following topics:

Determining the Global VCAT and BSDS Size (see page 31)
Determining the Cache Size (see page 32)
Deciding on a Cache Management Strategy (see page 34)
Dynamic Cache Management (see page 35)
Static Cache Management (see page 41)

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.

Determining the Cache Size

If CA Sales Consulting has already performed a tape study to determine the appropriate cache size, you can proceed to the section Deciding on a Cache Management Strategy (see page 34). 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 sizing the DASD buffer (cache), 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. 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.

Deciding on a Cache Management Strategy

CA Vtape uses VSAM Linear Data Sets (LDSs) to represent Virtual Volumes. Two types of cache management are provided:

- Dynamic
- Static

Dynamic is the newest way to manage the LDSs, introduced in r11.5. When Dynamic Cache Management is active, CA Vtape dynamically defines a cache LDS when it is needed. SMS controls the allocation of the LDS. The LDS can be idle-space released, take secondary extents, and span to additional DASD volumes. A single LDS is defined for a Virtual Volume and the data set name of the LDS contains the Virtual VOLSER. The final size of the LDS is essentially determined by the amount of application data written to the Virtual Volume or the defined maximum Virtual Volume Size.

A Cache Monitor periodically scans the SMS storage group or groups and deletes externalized Virtual Volumes to maintain a certain percentage of free space. The percentage of free space is defined by you in the CA Vtape parmlib. The monitor also automatically releases and holds the Externalization Subgroup Queues based on customizable parmlib attributes.

The dynamic LDSs are accessed using SMS Media Manager. Media Manager reduces the amount of CPU used by CA Vtape for data movement by 50%. Media Manager also provides 50% higher throughput.

With Dynamic Cache Management multiple Virtual Volume Pools can be defined and each pool can have its own Virtual Volume Size defined of 400, 800, 2000, 4000, 8000, or 16000 MBs.

The original way to manage the LDSs is referred to as Static Cache Management. The cache LDSs are predefined and initialized to the same size and added to CA Vtape with batch jobs built by the ISPF Customization Process. Each LDS is 1/10th the size of the uncompressed capacity of the Virtual Volume Size you choose during the ISPF Customization Process. SMS management is optional. The LDSs cannot be idle-spaced released, cannot take secondary extents, and cannot be spanned to additional DASD volumes. From one to ten of these LDSs make up one Virtual Volume depending on the amount of application data written. To determine which Virtual VOLSER resides in which static LDS, you must run a batch report.

The Cache Monitor only automatically releases and holds the Externalization Subgroup Queues. The cache is not monitored for free space because the LDSs are never deleted; they are reassigned to a new VOLSER and reused.

The static LDSs are accessed using Data-In-Virtual (DIV). DIV provides good performance, but is CPU-intensive.

Static Cache Management does not support subpooling of the Virtual Volumes and those volume must be defined as 400, 800, or 2000 MBs.

The following table shows a comparison of Static and Dynamic characteristics at a glance:

Characteristics	Static	Dynamic
Access Technique	DIV	Media Manager
CPU Usage	50% higher than Media Manager	50% less than DIV
Throughput	50% less than Media Manager	50% higher than DIV
DFSMS Managed	Your choice	Required
Cache LDS Allocation Method	Predefined and initialized with batch jobs	Defined as needed
Idle-Space Released	No	Yes
Secondary Extents	No	Yes
Span DASD Volumes	No	Yes
Cache Monitor	Reassigns LDSs during mount and recall processing	Maintains free space for immediate allocation
Virtual Volume Subpooling	No	Up to eight subpools
Virtual Volume Size	400, 800, or 200 MBs; only one size per Subsystem	400, 800, 2000, 4000, 8000, and 16000 MBs; each defined pool can be defined with a different size
LDS Size	All are 40 or 80 or 200 MBs	Varies from 12 to 2000 MBs as needed
LDSs per Virtual Volume	1-10	1
VOLSER in LDS name	No	Yes

Important! We highly recommend you use Dynamic Cache Management.

Dynamic Cache Management

This section provides the steps for establishing Dynamic Cache Management. If you want to convert from existing static cache to dynamic cache, see the section Migrating to Dynamic Cache Management in the appendix "Conversion Procedures" in the *Administration Guide*.

Note: If you decided that Static Cache Management will be used, you can proceed to the section Static Cache Management (see page 41).

Setting Up Dynamic Cache Management in SMS

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

Unique SMS constructs should be defined for use by CA Vtape. Defining unique constructs will provide better performance and greater control of the DASD cache volumes.

SMS Data Class

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

```
Volume Count . . . . . .: 16
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 . .: 59
```

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

The following explains each characteristic:

Volume Count

Reserves space in each cache LDS catalog entry for the number of volumes entered. 16 is recommended to reserve some space in the catalog entry for multiple volume allocations and to minimize the number of CI/CA splits versus the maximum of 59 which would eliminate all splits, but would require that a large catalog be defined.

Data Set Name Type

Extended ensures that the Virtual Volume Data Sets are allocated in extended format. This is required by partial space release, which is used to release unused DASD space allocated to a cache LDS.

Extended Addressability

Deals with data sets greater than 4 GB. If you intend to write Virtual Volumes of 8 GBs or 16 GBs, SMS requires that this be set to YES.

Record Access Bias

Allows CA Vtape to control record buffering.

Space Constraint Relief and Reduce Space Up To (%)

These parameters allow SMS to modify the space allocation requested by CA Vtape when the storage group becomes constrained for space.

Dynamic Volume Count

Allows the cache LDS 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.

SMS Storage Class

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

```
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 LDSs 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.

SMS Management Class

A management class does not need to be defined or assigned to the LDSs. The LDSs do not need to be backed up or migrated. Their life cycle is managed by CA Vtape.

Alternately, a do nothing management class can be defined and assigned to the LDSs. A do nothing management class would be one that has 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, and AUTO BACKUP set to NO.

SMS Storage Group

A unique storage group should be defined for the DASD buffer volumes. Creating a unique storage group makes it easier for both you and CA Vtape to manage DASD usage and monitor performance.

Since migration and backup of the Dynamic Cache LDS's is not required, the Auto Migrate, Auto Backup, and Auto Dump fields in the storage group should be set to no.

SMS ACS Routines

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

If the ACS routines are coded to honor the data class and storage class requested by a user, then no changes are required in the data class or storage class ACS routines. If the ACS routines are set up to automatically override or validate the users' requested data class and storage class, then modifications to these ACS routines will be required.

Because the storage group is assigned by the ACS routines, you must update the storage group ACS to assign the storage group defined for CA Vtape.

The life cycle of the cache LDSs is controlled by CA Vtape. These LDSs do not need to be backed up and should not be migrated. Therefore, you do not need to assign a management class to these LDSs. If the management class ACS routine automatically defines a management class to each allocation or assigns a default, changes may be required.

The sample ACS routine changes in this section assume that:

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

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

```
IF &DSN = CAVTAPE1.VVE.V*.MM.CACHE OR
   &DSN = CAVTAPE1.VVE.VDATAQ.** OR
   &DSN = CAVTAPE1.VVE.D*.* THEN DO
     SET &DATACLAS = 'DCVTAPE'
     EXIT CODE(0)
     END
```

Note: The data class, DCVTAPE in the sample above, is used by CA Vtape to define its DASD data sets. This data class should not be used in the SMS ACS routines to direct tape mounts to CA Vtape. A separate data class or classes should be defined for intercepting tape mounts. If a DASD data class is used to intercept tape mounts, the tape mounts will be converted by SMS to DASD allocations and the allocations will fail, typically with a SPACE error.

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

- The cache LDSs that contain the application data.
- 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.

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

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

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

DASD Allocation Control Products

The CA Vtape dynamic LDS allocations requested by the SVTS and SVTSAS started tasks should not be intercepted or modified by DASD allocation control software other than SMS. Products which intercept space abends or adjust allocation amounts based on DASD pool conditions are not needed since Space Constraint Relief is coded in the SMS data class. Also, CA Vtape requests specific primary and secondary space values based on the CachePrimary and CacheSecondary attributes coded in its parmlib 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. The CA Allocate ASR should be updated to ignore all CA Vtape dynamic cache allocations.

Dynamic Cache Virtual Volume Size

For Dynamic Cache Management the Virtual Volume Size chosen determines the point at which CA Vtape will force the tape allocation to another Virtual Volume creating a multiple volume tape data set. This is accomplished by presenting an End-Of-Volume condition.

We recommend you use 2000 MB Virtual Volumes. This allows the cache LDSs to vary in size as needed from a minimum of 12 MBs for very small data sets to as large as 2000 MBs for large data sets. This is the most efficient way to use space in the storage group and minimizes the number of multivolume virtual tape data sets created.

Virtual Volume Sizes of 4000, 8000, and 16000 MBs can be coded with the VolumeSize attribrute in the Volume Pool Definitions Section of the CA Vtape parmlib.

Static Cache Management

This section provides the steps for establishing Static Cache Management.

If you want to convert from existing static cache to dynamic cache, see the section Migrating to Dynamic Cache Management in the appendix "Conversion Procedures" in the *Administration Guide*.

If you want to revert back to Static Cache Management mode, see the section Return to Static Cache Management Mode in the chapter "Conversion Procedures" in the *Administration Guide*.

Static Cache Virtual Volume Size

Virtual Volumes are defined during customization as 400 MBs, 800 MBs, or 2000 MBs.

For Static Cache Management, the Virtual Volume Size chosen determines the size of the LDSs that must be defined and initialized on the cache volumes that CA Vtape uses. Each LDS is 1/10th the size of the Virtual Volume Size you choose. One LDS is the smallest unit of DASD allocation for a Virtual Volume. Ten LDSs make up a full Virtual Volume.

400 MB Virtual Volumes are best for sites planning to use CA Vtape for small physical tape data sets to take advantage of automatic stacking. The 40 MB cache LDSs defined to support this Virtual Volume size maximize the number of small tape data sets that can be stored in the cache at one time and minimizes the overall size of the cache. The average uncompressed size of the physical tape data sets should be 400 MBs or less.

800 MB Virtual Volumes are best for sites planning to use CA Vtape for small physical tape data sets to take advantage of automatic stacking and larger physical tape data sets that might fill a physical 3490E tape. The 80 MB cache LDSs defined to support this Virtual Volume size offer a compromise between the most efficient use of the cache DASD space and limiting the increase in the number of Virtual Volumes per data set, which might require the coding of a volume count parameter in the application JCL. A volume count parameter is required whenever more than five tape volumes are required to hold a data set. The average uncompressed size of the physical tape data sets should be less than 800 MBs.

2000 MB Virtual Volumes are best for sites planning to use CA Vtape for physical tape data sets that fill a physical 3490E or larger tape. The 200 MB cache LDSs defined to support this Virtual Volume size provide the maximum relief from the increase in the number of Virtual Volumes that are required to hold these large tape data sets. This minimizes the need to add a volume count parameter to the application JCL.

You can run multiple CA Vtape subsystems on the same LPAR. This allows you to define for example, an 800 MB and a 2000 MB CA Vtape complex on the same LPAR. The CA Vtape filters can then be used to route smaller physical tape data sets to the 800 MB complex and larger physical tape data sets to the 2000 MB complex. While this increases the complexity of the CA Vtape installation and filters, it more efficiently uses the cache DASD space assigned to each complex, maximizes the number of small and large data sets that can reside in the two caches, and minimizes the need to add a volume count parameter to the application JCL.

Determining the Number of Static Cache DASD Volumes

The effective capacity for the various DASD device types is less than the total DASD device capacity. What follows is a breakdown for each device type, for the 400, 800, and 2000 MB Virtual Volume sizes. Assumptions for all of the device types are that the devices contain no other data, and that two cylinders contain the VTOC, VTOC Index, and the VVDS.

To calculate the number of DASD volumes needed for the forecasted DASD buffer (cache), use the following formula:

```
(DASD buffer size * 1000 / LDS size) / LDSs per Volume
```

The LDS size for 400 MB Virtual Volumes is 40 MBs. The LDS size for 800 MB Virtual Volumes is 80 MBs. The LDS size for 2000 MB Virtual Volumes is 200 MBs.

The following table lists LDSs per volume for 400 MB Virtual Volumes. Each LDS will use 854 tracks:

Device Type	LDSs per Volume
3390-3	58
3390-9	175

The following table lists LDSs per volume for 800 MB Virtual Volumes. Each LDS will use 1707 tracks:

Device Type	Maximum LDS
3390-3	29
3390-9	88

The following table lists LDSs per volume for 2000 MB Virtual Volumes. Each LDS will use 4267 tracks:

Device Type	Maximum LDS
3390-3	11
3390-9	34

For example, a 100-GB DASD buffer with 400-MB Virtual Volumes allocated on 3390-3 devices would generate the following formula and answer:

(100 * 1000 / 40) / 58 = 43.1 (rounded to 44)

A total of 44 3390-3 DASD volumes would be required.

Setting up Static Cache Management

For Static Cache Management, the cache may or may not be managed with SMS. The only benefit of using SMS or another allocation management software is that it eliminates the need to enter a list of DASD VOLSERs into a panel during the customization process or when expanding the DASD buffer at a later time.

SMS Managed Cache

For a SMS-managed cache, you must define a SMS storage class and storage group containing the appropriate number of DASD volumes you calculated in the section Determining the Number of Static Cache DASD Volumes. The storage class name is used during the customization process to generate the appropriate IDCAMS DEFINE statements to create the static cache LDSs. These LDSs will have a cluster name of PREFIX.VVE.LDS*nnnnn* and a data component name of PREFIX.VVE.LDS*nnnnn*.DATA where *nnnnn* is a number between 00001 and 65535.

The sample ACS routine changes all assume the following:

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

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

```
IF &DSN = CAVTAPE1.VVE.LDS* THEN DO
     SET &STORCLAS = 'SCVTAPE'
     EXIT CODE(0)
     END
```

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

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

When using SMS to manage the allocation of the static cache LDSs, a management class that backs up or migrates the cache LDSs should not be assigned to the allocations. These LDSs need to remain on DASD where CA Vtape can access them immediately. Migrating them will cause virtual mount delays.

Alternately, a do nothing management class can be defined and assigned to the LDSs. A do nothing management class would be one that has 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, and AUTO BACKUP set to NO.

Non-SMS Managed Cache

For a non-SMS managed DASD buffer, a list of DASD VOLSERs will be entered during the customization process. Alternatively, you can use an allocation management software like CA Allocate to define a storage pool. In this case, you only enter one Volume Serial during the customization process.

Chapter 4: ISPF Customization Panels

This section contains the following topics:

Product Customization Panels (see page 45)
ISPF Customization Panel Primary Options (see page 46)
ISPF Customization Steps (see page 47)

Product Customization Panels

The Customization Process uses input from ISPF panels to create JCL members and a configuration member SUTPARMS in the PREFIX.SVTJCL library. These members are then used to define and initialize the global or shared resources in a CA Vtape Complex such as the Global VCAT and BSDS control data sets.

The SUTPARMS member contains the variables for the DSN prefix and the Virtual Volume Size. This member is used to initialize the Local VCAT.

The ISPF panels include help screens and a tutorial.

If the Customization Process is executed multiple times, deleting and generating the JCL and configuration members, the SVTJCL library may need to be compressed between executions or reallocated to a larger size.

ISPF Customization Panel Primary Options

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

(V)iew

View CA Vtape Customization Log. This option allows the viewing of the CA Vtape customization options log. The customization log is rewritten every time a customization session is completed. The customization log documents the parameter values entered during the last customization session and statistics concerning the JCL generated.

Copy the CA Vtape customization log to a protected data set to maintain a historical reference of past customization sessions.

(R)egenerate

Regenerate JCL for an existing CA Vtape complex. This option regenerates the SVTJCL library JCL members and the SUTPARMS configuration member. You cannot change the previously entered DSN Prefix and Virtual Volume Size. Changing these values would make the JCL and SUTPARMS members generated incompatible with an existing CA Vtape Complex. These values can only be changed with the Define option.

Invoke this option to regenerate the installation JCL to replace lost JCL and SUTPARMS members.

(D)efine

Define a new CA Vtape complex. This option generates the SVTJCL library JCL members and the SUTPARMS configuration member used to define and initialize a CA Vtape Complex. The JCL includes delete statements for an existing CA Vtape Complex. Changing the CA Vtape DSN Prefix and Virtual Volume size are 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 elements of an existing CA Vtape complex are deleted and 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.

ISPF Customization Steps

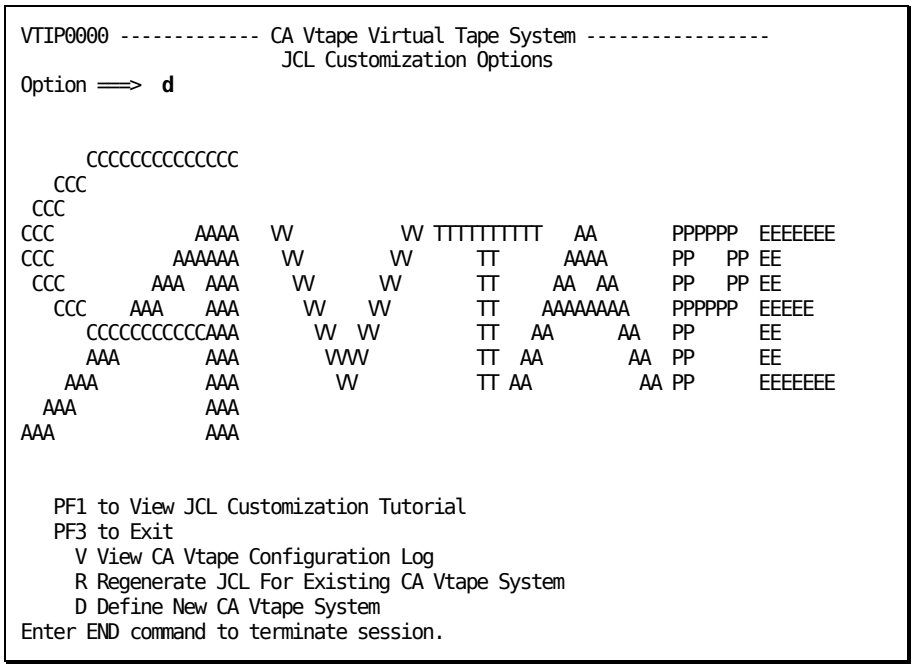
Perform the following steps to complete the ISPF Customization Process.

1. Type the Following Command in ISPF option 6:

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

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 CA Vtape main menu.



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

```
VTIP1000  --------------- CA Vtape Virtual Tape System ----------------
                          JCL Customization Options
COMMAND ===>  c                                          C to Continue
                                                        PF3 to Previous


  Specify The SVTJCL Data Set Name (Prefix.SVTJCL)

    SVTJCL Library: CAVTAPE1.SVTJCL_____________________________

  The CA Vtape SVTJCL data set is the repository for the
  generated customization JCL, the Configuration Log, and the
  Configuration Member SUTPARMS.

***
```

4. Update as needed the generated jobcard which contains your user-id. Enter the Subsystem ID and APF authorized loadlib for CA Vtape. Enter **c** on the command line and press Enter to continue to the next panel.

```
VTIP1200  --------------- CA Vtape Virtual Tape System ----------------
                          JCL Customization Options
COMMAND ===>  c                                          C to Continue
                                                        PF3 to Previous


 JCL Job Statement Definition:

===> //useridA JOB (ACCOUNT),'CA VTAPE UTILITY',CLASS=A,_______________
===> //             MSGLEVEL=(1,1),MSGCLASS=X,REGION=4M,_______________
===> //             NOTIFY=userid_____________________________________


 Define the SVTS Subsystem ID. You may define SVT1 through SVT8.

 SVTS Subsystem ID:  ===> SVT1


 APF Authorized Load Library (HLQ.CCUULOAD):

   ===> CASMPE.VTAPE.CCUULOAD_____________________________________

***
```

5. Enter the CA Vtape DSN Prefix. The default value is CAVTAPE1. Enter **c** on the command line and press Enter to continue to the next panel.

```
VTIP130D  --------------- CA Vtape Virtual Tape System ----------------
                          JCL Customization Options
COMMAND ===>  c                                          C to Continue
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.


     CA Vtape System DSN Prefix:  CAVTAPE1_________________


     CA Vtape System Data Sets:
      * Prefix.GLOBAL.VCAT             Global Vcat Data Set
      * Prefix.BSDS1                   Bootstrap Data Set
      * Prefix.sysid.SVTnVCAT          Local Vcat Data Sets
      * Prefix.VVE.LDSnnnnn            Static DASD Buffer Data Sets
      * Prefix.VVE.Vvolser.MM.CACHE    Dynamic DASD Buffer Data Sets
      * Prefix.VVE.Vvolser.PRIMARY     Primary Backstore Data Sets
      * Prefix.VVE.Vvolser.DUPLEX      Duplex Backstore Data Sets
      * Prefix.SVTJCL                  System JCL Libary
      * Prefix.PARMLIB                 System Parameter Library

***
```

6. The Global VCAT and BSDS1 data set names are generated using the DSN Prefix entered on the previous panel. Enter the required allocation information. Enter **c** on the command line and press Enter to continue to the next panel.

   Note: If the Global VCAT and BSDS1 DSNs need to match a control data set naming standard at your site, they can be renamed in the JCL generated by the Customization Process before the JCL is submitted. CA Vtape only enforces the naming standard using the DSN Prefix during the Customization Process. The running product does not enforce this naming standard for these data sets. You will be reminded to rename these data sets if required during the appropriate step later in the Customization Process.

```
VTIP1400  --------------- CA Vtape Virtual Tape System ----------------
                          JCL Customization Options
COMMAND ===>  c                                          C to Continue
                                  P for Profile Values  PF3 to Previous


 DFSMS constructs and/or VOLSERs can be defined to control the
 allocation of the Global VCAT and BSDS1 data sets. The Global
 VCAT and BSDS1 should not be allocated on the same volume and
 should not be allocated on the DASD Buffer Pool volumes.

 Global VCAT Data Set Name: CAVTAPE1.GLOBAL.VCAT

                Data Class:  ____________________________________________
             Storage Class:  GVSTORCLAS_____________________________________
          Management Class:  ____________________________________________

                   VOLSER:  ___________

 BSDS1 Data Set Name: CAVTAPE1.BSDS1

                Data Class:  ____________________________________________
             Storage Class:  BSSTORCLAS_____________________________________
          Management Class:  ____________________________________________

                   VOLSER:  ___________

***
```

7. On this panel you chose whether you are going to customize CA Vtape to use Dynamic or Static Cache Management.

   The panel defaults to Dynamic Cache Management. To take the recommended path of Dynamic Cache Management, enter **c** on the command line and press Enter to continue to the next panel. Proceed to step 10.

   To take the Static Cache Management path, overtype DYNAMIC with STATIC, enter a c on the command line and press Enter to continue to the next panel.

```
VTIP145D  --------------- CA Vtape Virtual Tape System ----------------
                           JCL Customization Options
COMMAND ===>  c                                          C to Continue
                                    P for Profile Values  PF3 to Previous

 Define CA Vtape CACHE Management

  - Specify "Dynamic" or "Static"
    - Dynamic Cache Management dynamically allocates DFSMS managed
      data sets using Media Manager I/O Services.
    - Static Cache Management pre-allocates and pre-initializes a
      fixed number of VSAM Linear data sets using Data-In-Virtual I/O
      services. In this mode DFSMS is not required.

      CA Vtape Cache Management:  DYNAMIC

 ***
```

8. This panel is only displayed for Static Cache Management. If you chose Dynamic Cache Management, proceed to step 10.

   Enter the required allocation information for the DASD buffer LDSs. Enter **c** on the command line and press Enter to continue to the next panel.

```
VTIP1500  --------------  CA Vtape Virtual Tape System  --------------
                           JCL Customization Options
COMMAND ===>  c                  E to Edit Volumes Mbr   C to Continue
                                 P for Profile Values  PF3 to Previous


 DFSMS constructs or a volume list can be used to control the
 allocation of the DASD Buffer Pool Linear Data Sets. A Volume List
 Member can be defined in the SVTJCL library from this panel by
 specifying a member name that must start with VOL (e.g., VOLUMES,
VOLIST1). The DFSMS constructs and the Volume List Member are
 mutually exclusive.

           Data Class:  ________________________________
         Storage Class:  BPSTORCLAS_____________________
      Management Class:  ________________________________

    DASD Buffer Pool Volume List Member Name: __________

***
```

9. This panel is only displayed if you have chosen Static Cache Management. If you chose Dynamic Cache Management proceed to step 10.

   Enter the Virtual Volume Size, the total number of Virtual VOLSERs that will be needed in the CA Vtape Complex, and the DASD Buffer Pool size. The number of Virtual Volumes defaults to 1000. The number entered is used to size the control data sets. 1000 works well for a software trial. If a production complex is being defined, the value should be increased to at least 100,000. Enter **g** on the command line and press Enter to generate the customization JCL, the Configuration Log, and the Configuration Member SUTPARMS.

   Proceed to step 11.

```
VTIP160D  --------------  CA Vtape Virtual Tape System  --------------
                            JCL Customization Options
 COMMAND ===>  g                                          G to Generate
 Session Mode: Define                                     PF3 to Previous


    Virtual Tape Volume Size (MBytes):  800_

  A Virtual Tape Volume Size of 400, 800 or 2000 MBytes can be entered.
  The value entered is used to internally configure
  CA Vtape and to determine the size of the DASD Buffer Pool Linear
  Data Sets (LDS).

          400 MByte Virtual Volumes = 40 MByte LDS
          800 MByte Virtual Volumes = 80 MByte LDS
         2000 MByte Virtual Volumes = 200 MByte LDS

  Define the maximum number of Virtual Volumes for the
  CA Vtape System. Enter a value between 100 and 1,000,000

    Number of Virtual Volumes to Index:  100000__

  Enter the DASD Buffer Pool Size (MBytes) determined from the
  analysis of your tape activity reports. The maximum value
  for 400 MByte Virtual Volumes is 2,621,440 MBytes. The maximum
  value for 800 MByte Virtual Volumes is 5,242,880 MBytes.  The
  maximum value for 2000 MByte Virtual Volumes is 13,107,200 MBytes.

    DASD Buffer Pool Size (MBytes):      100000_____

 ***
```

10. This is the panel displayed for Dynamic Cache Management. If you chose Static Cache Management, proceed to step 11.

    The Virtual Volume Size defaults to 2000 MBs. This is the recommended value. Enter the total number of Virtual VOLSERs that will be needed in the CA Vtape Complex. The number entered is used to size the control data sets. 1,000 is the default and this works okay for a software trial. If a production complex is being defined, the value should be increased to at least 100,000. Enter **g** on the command line and press Enter to generate the customization JCL, the Configuration Log, and the Configuration Member SUTPARMS.

```
VTIP170D  --------------- CA Vtape Virtual Tape System ----------------
                          JCL Customization Options
COMMAND ===>  g                                          G to Generate
Session Mode: Define                                   PF3 to Previous


    Virtual Tape Volume Size (MBytes):  2000

 A Virtual Tape Volume Size of 400, 800 or 2000 MBytes can be entered.
 The value entered is used to internally configure CA Vtape
 and to determine the maximum DASD allocation allowed for a single
 Virtual Tape Volume. If additional DASD space is needed, the Virtual
 Tape Volume will be kept, a scratch mount will be requested, and the
 application data set will go to multiple volumes.

 Define the maximum number of Virtual Volumes for the
 CA Vtape System. Enter a value between 100 and 1,000,000.

    Number of Virtual Volumes to Index:  10000_
***
```

11. During the generation process the following screen will be displayed. The message block will change multiple times as different phases of the generation process are entered.

    When the generation process is complete, you will be placed in a browse session to review the Configuration Log.

```
VTIP1000  --------------- CA Vtape Virtual Tape System ----------------
                          JCL Customization Options
COMMAND ===>                                             C to Continue
                                                       PF3 to Previous


   Specify The SVTJCL Data Set Name (Prefix.SVTJCL)

      SVTJCL Library:  CAVTAPE1.SVTJCL

                  ____________________________________________________
                 |                                                    |
   The CA Vtape Syst| Please wait ... Deleting Previously Defined SVTJCL |
   generated custom | Library Members                                    |
   Configuration ME |____________________________________________________|
  ***
```

12. After reviewing the Configuration Log press PF3 and you will be placed in an edit session for the PREFIX.SVTJCL(DELETE) member.

13. The DELETE JCL member submits other jobs. As the member name implies, the other jobs contain delete statements.

    Important! If you enter the DSN Prefix of an existing CA Vtape Complex into the panels and submit the DELETE JCL member, you run the risk of deleting the control data sets and the cache and backstore copies of the Virtual Volumes of that existing CA Vtape Complex. That is, you run the risk of creating a data loss situation.

14. Press PF3 to return to the Main Menu. Press PF3 again to leave the Customization Panels.

Chapter 5: Control and Cache Data Sets

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

This section contains the following topics:

Control Data Sets (see page 55)
Define the Global VCAT and BSDS (see page 56)
Define the Local VCATs (see page 56)
Define the Cache Data Sets (see page 57)

Control Data Sets

The control data sets consist of the following:

Local VCAT

This file records information specific to a single CA Vtape Subsystem. Each Subsystem must have its own unique Local VCAT. The Local VCAT is used primarily as a work area for transient information and to contain the attribute settings loaded from parmlib. Since it is a work file, it is not backed up and restored for disaster recovery. Instead, a new one is defined.

Global VCAT

This file records multi-system control information such as Virtual Volume status. This file is shared by all of the Subsystems participating in a CA Vtape Complex. Since it is mirrored by the BSDS1, the Global VCAT is not backed up and restored for disaster recovery. Instead, it is recovered from the BSDS1.

BSDS1

This file records information necessary to recover the Global VCAT. Like the Global VCAT, this file is shared by all of the Subsystems participating in a CA Vtape Complex. The BSDS1 is backed up for disaster recovery.

The control data sets should be considered critical system files. Keeping these files on high performance, reliable media, such as a RAID-compliant DASD, will help to improve overall performance and minimize potential system outages. Maintaining current backups of the BSDS1 and keeping the Global VCAT and BSDS1 on separate DASD volumes is critical to insuring recoverability if the hardware unexpectedly fails.

A Hardware Reserve is used processing to serialize access to the Global VCAT. The Hardware Reserve may require you to isolate this data set on a single DASD volume.

Define the Global VCAT and BSDS

Follow these steps:

1. Edit the PREFIX.SVTJCL library member named GLOBAL.

   If the control data sets created by the GLOBAL JCL member need to be renamed to match a site naming standard, rename them now. You can also make general changes to the volume parameters, SMS construct names, and the job cards.

   If you need to change the DSN Prefix or the Virtual Volume Size, you should execute the Customization Panels again and enter the new values so these values can be properly built into the SUTPARMS member. The SUTPARMS member is used to initialize the control data sets.

2. When your review is complete, save the member.
3. Edit the PREFIX.SVTJCL library member named DELETE.

   The DELETE JCL member may contain delete statements and submit other jobs which contain delete statements.

   Important! If the DSN Prefix you entered into the Customization Panels was for an existing CA Vtape Complex, that DSN prefix was built into the DELETE statements in these jobs. If you submit the DELETE JCL member, you will delete the control data sets and the cache and backstore copies of the Virtual Volumes of that existing CA Vtape Complex. That is, you run the risk of creating a data loss situation.

4. When your review is complete, submit the Delete member.
5. Review the job output and the output of any additional jobs that were automatically submitted. Verify that all job steps completed successfully. If any of the jobs failed, take appropriate 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.

Define the Cache Data Sets

If you chose Dynamic Cache Management as recommended, there are no cache data sets to define.

If you chose Static Cache Management, submit the PREFIX.SVTJCL member SUBLDSD. This member will in turn submit all the LDSDEFxx jobs built to define the static cache LDSs.

Since the static cache LDSs must be formatted before they can be used, these jobs will run for an extended period of time. If any of the jobs end with a non-zero return code, review the job output to determine why, correct the error, and resubmit the job.

Chapter 6: Tape Mount Intercept Filters

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

This section contains the following topics:

General Description (see page 59)
Tape Mount Redirection (see page 59)
Tape Mount Intercept Filters (see page 62)
VTFILTR Parmlib Member (see page 62)
Date Set Name Pattern Masking (see page 63)
Filter Processing (see page 64)

General Description

For CA Vtape to intercept tape mounts, Data Class or Data Set Name Filters must be coded in the filter member of the CA Vtape parmlib. The default parmlib member name is VTFILTR.

The coded filters are sorted and loaded into the Local VCAT when the CA Vtape Subsystem is started. If the filter member is modified while the Subsystem is active, the SVT*n* REFRESH=FiLTer console command must be issued for each Subsystem that is sharing the filter member to reload the filters.

Tape Mount Redirection

SMS data class constructs and data set names may be used to redirect tape mounts to CA Vtape Virtual Devices. Each technique has advantages and disadvantages.

Using SMS Data Class Constructs

SMS Data Class names can be entered into a CA Vtape parmlib member to redirect a tape mount to a Virtual Device. Using a data class does not mean the data set is DFSMS-managed.

The advantages of this technique are the opportunity to use the many selection criteria available to SMS, such as DSN, UNIT, JOBNAME, and USERID and the centralization in SMS 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 SMS and ACS routines, see the IBM *DFSMSdfp Storage Administration* Reference.

Using Data Set Name Filtering

Data set names or patterns can be entered into a CA Vtape parmlib member to redirect a tape mount to a Virtual Device.

The advantages of this technique are 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 as needed.

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

Using Both SMS Data Class Constructs and Data Set Name Filtering

CA Vtape Data Class Filter List matching takes precedence over the Data Set Name Pattern Filter List. 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 recommended using both techniques.

Using 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 code indicating that a tape subsystem had 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.

CA Vtape ignores the filter match and allows the tape mount to be serviced by whatever devices are referenced by the esoteric or generic used to avoid the delays caused by job or task abends. 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 by CA Vtape. However, if UNIT=3590 or UNIT=TAPE was used for data set PROD.MASTER.FILE, CA Vtape 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.

VTFILTR Parmlib Member

The Parmlib Directory Section in the VTPARMS parmlib member contains attributes which tell CA Vtape where specific sets of related parmlib attributes are located. One of the attributes is DatasetFilters which has a default value of VTFILTR. The VTFILTR parmlib member is where you code your tape mount intercept filters.

VTFILTR 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 contain include or exclude filters.

The index section name is <DatasetFilters>. In this section you code IncludeDatasets, ExcludeDatasets, and IncludeDataClass attributes. If you are licensed for the P2P Option, you can also code RemoteIncludeDatasets, RemoteExcludeDatasets, and RemoteIncludeDataClass attributes. These attributes can be coded as many times as needed to properly organize and document your filter sections. The values of these attributes are unique, customized section names which contain filters of the type 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 will use 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 VTGROUP parmlib member. Groups are used like DASD storage groups or pools to associate tape data sets with similar storage management requirements. All production tape data sets could be assigned to group 01, all test tape data sets could be assigned to group 02, and all disaster recovery tape data sets could be assigned to group 03 allowing you to manage these groups of data 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 set up:

```
<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./
```

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

Note: The SVT*n* REFRESH=FiLTers console command is local in scope. If three CA Vtape subsystems running on two Logical Partitions are using the same filter member that was changed, the console command must be issued 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 called 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 pattern DEPT751* finds all data sets that start with DEPT751. To obtain the same result in CA Vtape, use the pattern DEPT751/.

!

When this character (English exclamation mark, X"5A") is encountered, the input is searched for a match on the characters that follow it. 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 using this <ParmlibDirectory> section, the following startup messages are displayed:

```
IEE252I MEMBER  VTPARMS FOUND IN SVTSDEMO.VTAPE.XE21.PARMLIB
IEE252I MEMBER  VTPARMS FOUND IN SVTSDEMO.VTAPE.XE21.PARMLIB
IEE252I MEMBER  VTPARMS FOUND IN SVTSDEMO.VTAPE.XE21.PARMLIB
IEE252I MEMBER  VTDRIVE FOUND IN SVTSDEMO.VTAPE.XE21.PARMLIB
IEE252I MEMBER  VTGROUP FOUND IN SVTSDEMO.VTAPE.XE21.PARMLIB
IEE252I MEMBER  VTFILTR FOUND IN QAPROD.VTAPE.SMP120.PARMLIB

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

A warning message is issued if no include filter entries are found. 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 SMS 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 7: Scratch Tape Synchronization

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 (see page 67)
Scratch Tape Synchronization (see page 67)
CA 1 Tape Management System (see page 68)
CA TLMS Tape Management (see page 70)
DFSMSrmm (see page 70)
Control-T (BMC) (see page 70)
AutoMedia (Zara) (see page 70)
Other Tape Management Systems (see page 71)
Scratch Pool Processing (see page 71)
CA Vtape P2P Option Scratch Pool Processing (see page 73)

Tape Expiration

The installed tape management system controls the expiration of the Virtual Volumes and the Backstore Tapes. The Virtual Volumes will be under whatever tape retention control that the application requested in their JCL or dynamic allocation request, or the retention specified by tape management system retention rules. The Backstore Tapes should be under catalog control retention and not expired until all data sets on the tape have expired.

When the Virtual Volumes or Backstore Tapes 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. Backstore Tape expirations do not need to be communicated to CA Vtape.

Scratch Tape Synchronization

Expiration of the CA Vtape Virtual Volumes is controlled by the Tape Management System. As Virtual Volumes are expired, CA Vtape must be notified for the following actions:

- Changing the status of the Virtual Volumes in the Global VCAT.
- Deletion of the cache copies of the Virtual Volumes.
- Uncataloging of the backstore copies of the Virtual Volumes.

- Making scratched Virtual Volumes available for reuse.
- Notify the appropriate remote CA Vtape Subsystem to scratch shared VOLSERs (CA Vtape P2P Option only).

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.

If a CA Vtape Complex has been created, multiple CA Vtape Subsystems running on the same or different LPARs sharing the same Global VCAT and Tape Management System Database, the Scratch Synchronization Job only needs to be run once for the complex.

Important! When multiple CA Vtape Complexes are installed, you should execute all steps of the appropriate Scratch Synchronization Job, in the order provided, once for each Complex. If you do not, you could deplete all available scratch tapes or scratch active Virtual Volumes, causing a data loss.

The following sections of this chapter 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 68).
- The Scratch Synchronization Job (see page 69).

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 and 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 and the prefix character or characters of the defined Virtual Volume VOLSER ranges.

If you choose to use the CA 1 ROBSCR option you should 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. The issue being that TMSCLEAN will only notify CA Vtape when a Virtual Volume is scratched. If that notification fails, TMSCLEAN will 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 they are in-sync.

CA 1 DSNB Records

Review your data set name block (DSNB) counts to determine if enough DSNBs exist to support a large increase due to data set stacking. The CA Vtape Externalization process stacks Virtual Volumes onto Backstore Tapes. CA 1 consumes one DSNB for each Virtual Volume that is stacked starting with the second Virtual Volume stacked. If 100 Virtual Volumes are stacked on a Backstore Tape, 99 DSNBs will be consumed.

DSNBs exist for the life of the Backstore Tape, not the life of the Virtual Volume. As long as one of the Virtual Volumes on a Backstore Tape containing 100 Virtual Volumes remains unexpired, all 99 DSNBs are maintained, including the expired ones. As CA Vtape usage increases over time, more DSNBs will be used and maintained for expired Virtual Volumes. If all DSNBs are consumed, tape processing stops until the Tape Management Catalog (TMC) is increased in size and the number of DSNBs is increased.

Running the RECYCLE utility to recover fragmented Backstore Tape space will also recover expired DSNBs. However, a large increase in the number of DSNBs will still occur. A good rule is to have three DSNBs available for each Virtual Volume defined. If 50,000 Virtual Volumes were defined to CA 1, 150,000 free DSNBs should be made available.

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.

Scratch Pool Processing

Part of the function of the Scratch Synchronization Process is to return scratched Virtual Volumes to the Scratch Pool so they can be reused.

You can influence how scratch VOLSERs are chosen by CA Vtape by how you define the Virtual VOLSERs to CA Vtape. VOLSERs can be defined in the following ways:

- Parmlib member VTPOOL (see page 71)
- SVT*n* ADD VVP console command (see page 72)

We recommend you use the VTPOOL parmlib member.

VTPOOL Parmlib Member

When the VTPOOL member is being used to define the Virtual VOLSER range or ranges, newly scratched Virtual VOLSERs are immediately returned to the Scratch Pool and assigned a reuse date. CA Vtape will not reuse these VOLSERs until five days have passed or no other scratch VOLSERs are available. If enough of an appropriate number of VOLSERs are defined, this provides a five day recovery window before an accidentally scratched Virtual VOLSER is reused.

If the defined VOLSER ranges are broken up into separate subpools, a Scratch Pool is maintained for each subpool. Newly scratched Virtual VOLSERS are returned to the appropriate Scratch Pool and assigned a reuse date.

Each Scratch Pool is broken up into extents that are activated and deactivated individually. An extent consists of a set of 100 consecutive VOLSERs from *xxxx*00-*xxxx*99. When all scratch VOLSERs with a scratch date older than five days are depleted in an extent, the extent is deactivated and the next extent is activated. If all scratch VOLSERs older than five days are depleted in all extents, a second pass is made for the first available scratch VOLSER. The deactivation and activation of extents causes the Virtual VOLSERs to be chosen in order from lowest to highest in a round-robin fashion.

In the following example, V00006 is currently the next available scratch VOLSER in the currently active extent 3. If the Scratch Synchronization Job runs and makes V00001, V00002, and V00004 available for reuse, the active extent and the next available scratch pointer will not be changed. V00006 will still be the next available VOLSER. The next scratch mount request will use V00006 and turn it off. V00007 will then be the next available VOLSER. When V00007 is used, extent 3 will be deactivated and extent 1 will be activated.

```
BEFORE SYNCH    AFTER SYNCH     AFTER NEXT MOUNT
V00000          V00000          V00000           Extent 1*
V00001          V00001 On       V00001 On
V00002          V00002 On       V00002 On
V00003          V00003          V00003           Extent 2*
V00004          V00004 On       V00004 On
V00005          V00005          V00005
V00006 On <-    V00006 On <-    V00006           Extent 3* <-
V00007 On       V00007 On       V00007 On <-
V00008          V00008          V00008
* The extents were artificially shortened to three VOLSERs.
```

Note: When Virtual Volume Pooling is active, the SVT*n* ADD VVP and DELETE VVP console commands are deactivated. Changes to the Virtual VOLSER ranges are performed by updating the VTPOOL member and issuing the SVT*n* REFRESH=POOLS console command.

SVTn ADD VVP Console Command

If the VTPOOL parmlib member is not currently being used to define Virtual Volumes ranges, available scratch VOLSERs are maintained in a single Scratch Pool. The Scratch Synchronization Job makes two passes through the Virtual VOLSERs in the Global VCAT.

The first pass reads the Virtual Volume Entries in the Global VCAT and checks their matching entry in the Scratch Pool. If an entry is not active, it is updated to make it active which makes the VOLSER available for reuse. As entries are updated, the pointer to the next available scratch VOLSER is automatically updated.

The automatic pointer update causes the Virtual VOLSERs to always be reused in order from first to last. As scratch VOLSERs are used throughout the day, the pointer moves down through the pool. When the Scratch Synchronization Job runs, if it makes a previous entry available in the pool, the pointer is reset back to that entry. This entry will be the entry used to satisfy the next scratch mount request.

In the following example, V00006 is currently the next available scratch VOLSER. If the Scratch Synchronization Job runs and makes V00001, V00002, and V00004 available for reuse, the pointer will be reset to V00001. The next scratch mount request will reuse V00001 and turn it off. V00002 will then be the next available VOLSER.

```
BEFORE SYNCH     AFTER SYNCH      AFTER NEXT MOUNT
 V00000           V00000           V00000
 V00001           V00001 On <-     V00001
 V00002           V00002 On        V00002 On <-
 V00003           V00003           V00003
 V00004           V00004 On        V00004 On
 V00005           V00005           V00005
 V00006 On <-     V00006 On        V00006 On
```

Due to the pointer reset, the first entries in the Scratch Pool will always be reused before the last entries are ever used. If the Virtual VOLSER range is V00000-V99999, V00000 will be reused many times and V99999 might never be used.

The second pass uses a list of all Virtual Volumes in scratch status in the Tape Management System to update only the Virtual Volume Entries in the Global VCAT that are in active status. These newly scratched Virtual Volumes are not activated in the Scratch Pool as that is the function of the first pass and it has already been executed. These Virtual Volumes will be activated in the Scratch Pool by the next execution of the Scratch Synchronization Job. If the job is being run once every 24 hours, you have that amount of time to recover accidentally scratched Virtual Volumes before they eligible for reuse.

CA Vtape P2P Option Scratch Pool Processing

When the CA Vtape P2P Option is active, the Virtual VOLSERs should be defined in the VTPOOL member. The Scratch Synchronization Job will process the same as documented in the 'With Virtual Volume Pooling' section with one additional consideration.

Virtual Volumes created in a P2P configuration are owned by the CA Vtape Complex that first mounted them. These Virtual Volumes can only be scratched by their owning Complex. Attempting to scratch these volumes in another complex will result in a signature error.

A signature is an internal mechanism used by a CA Vtape Complex to identify a specific CA Vtape Complex. Each Virtual Volume has an owning signature and a remote signature field. Only the CA Vtape Complex with a signature matching the owning signature stored in the Virtual Volume Entry in the Global VCAT can scratch that Virtual Volume.

When the Scratch Synchronization Job executes, it will create a list of Virtual Volumes that were scratched that have a remote signature in their Virtual Volume Entries in the Global VCAT. The presence of a remote signature indicates that these Virtual Volumes were shared with a remote CA Vtape Complex. When the scratch process is complete, this list will be processed and broken down by remote CA Vtape Subsystem. These VOLSER subsets will then be transmitted to the appropriate remote CA Vtape Subsystem so it can synchronize its shared VOLSERs with the local or owning CA Vtape Subsystem.

Additional scratch related errors are documented by the SVTSDQ911E error message.

Chapter 8: Tape Management Systems

This chapter describes tape management system considerations other than those related to scratch synchronization.

This section contains the following topics:

Stacked or Multiple-File Tape Support (see page 75)
Tape Management System Interface (see page 77)
Foreign Tape Processing (see page 78)

Stacked or Multiple-File Tape Support

CA Vtape initially writes the Virtual Volumes to DASD. The Virtual Volumes are then stacked by the Backstore Engine onto Backstore Tapes. These Virtual Volumes are cataloged to the Backstore Tapes with a generated data set name of *prefix*.VVE.Vvolser.PRIMARY / DUPLEX, where *prefix* is the CA Vtape DSN prefix you defined for CA Vtape use.

Stacked or multiple-file physical tapes are supported in different ways by different tape management systems. The typical method is to use catalog control, which is automatically requested by the Backstore Engine when stacking.

Note: Catalog Control is requested by setting the expiration date in the dynamic allocation to a value of 99000. If your tape management system uses a different value, update the CatalogManagedDate attribute in the Dynamic Options Section of parmlib member VTPARMS with the appropriate date value. Issue the SVT*n* REFRESH=OPTIONS console command for each CA Vtape Subsystem that should pick up the change. Issue the SVT*n* Display Parmlib console command to verify that the date has been set correctly.

See the following sections for information about the specific tape management system installed.

CA 1 Tape Management System

CA 1 automatically supports stacked or multiple-file tapes by honoring the catalog control retention requested by the Backstore Engine. When the first Virtual Volume stacked on a Backstore Tape expires, the physical tape will not be expired. Instead, the Backstore Tape is maintained until all Virtual Volumes on the tape expire.

CA TLMS Tape Management

CA TLMS supports stacked or multiple-file tapes by assigning one of the following retention types:

- Retention Type: A, where all files must be uncataloged before a tape volume can be scratched.
- Retention Type: B, where all files must be expired before a tape volume can be scratched.
- Retention Type: C, which is a combination of A and B.

For CA Vtape, you should code a retention rule assigning type **A** retention to all tapes that contain data set names starting with "*prefix*.VVE" where *prefix* is the CA Vtape DSN prefix you defined for CA Vtape use.

Note: For details on coding retention rules, see the current *CA TLMS Tape Management User Guide* .

DFSMSrmm

RMM supports stacked or multiple-file tapes by using catalog control. Define a retention rule for all the CA Vtape Backstore Tapes using the RMM command ADDVRS DSNAME('*prefix*.VVE.V**') WHILECATALOG, where *prefix* is the DSN prefix you defined for CA Vtape use.

Review the other expiration rules to ensure that only catalog control expiration will be used for these tapes.

Control-T (BMC)

Control-T automatically supports stacked or multiple-file tapes by honoring the catalog control retention requested by the Backstore Engine. When the first Virtual Volume stacked on a Backstore Tape expires, the physical tape will not be expired. Instead, the Backstore Tape is maintained until all Virtual Volumes on the physical tape expire.

AutoMedia (Zara)

AutoMedia automatically supports stacked or multiple-file tapes by honoring the catalog control retention requested by the Backstore Engine. When the first Virtual Volume stacked on a Backstore Tape expires, the tape will not be expired. Instead, the Backstore Tape is retained until all Virtual Volumes on the tape expire.

CA Vtape by default uses the special expiration date of 99000 to indicate that a tape data set should be under catalog control. AutoMedia does not recognize this date. Instead, AutoMedia uses 00000 for the catalog control date. Update the CatalogManagedDate attribute in the Dynamic Options Section of parmlib member VTPARMS to change CA Vtape to this date. Issue the SVT*n* Refresh=Options console command for each CA Vtape Subsystem that should pick up the change. Issue the SVTn Display Parmlib console command to verify that the date has been set correctly.

Tape Management System Interface

CA Vtape supports an internal interface between CA 1 or CA TLMS. This interface retrieves Virtual Volume expiration date information from the Tape Management System and updates the Virtual VOLSER records in the Tape Management System with their stacking location (which Backstore Tape they were Externalized or Recycled to).

The TapeManagementSystem attribute in the Dynamic Options Section of the parmlib determines if the interface is activated.

Virtual Volume Expiration Date and Automatic Subgroup Adjustment

Each CA Vtape Externalization Group defined in the Group Definitions is automatically broken down into three subgroups, Short, Medium, and Long. Virtual Volumes are assigned to a subgroup based on the settings of the ShortRetention, MediumRetention, CatalogManagedSubgroup, NeverExpireSubgroup, and SpecialRetentionSubgroup attributes. The purpose of these subgroups is to break the Virtual Volumes in an Externalization Group up into subsets with common retention periods so that a Backstore Tape can expire without needing to be Recycled.

By default subgroup assignment is controlled by the expiration date or retention period coded in the JCL or dynamic allocation which first mounted the Virtual Volume. While this approach works well for most sites, it does not take into consideration that Tape Management Systems often have retention rules that override the retention coded in JCL. It also does not take into consideration that occasionally tapes have their expiration dates manually extended or shortened.

Activating the interface allows CA Vtape to retrieve the Virtual Volume expiration date from the Tape Management System when Externalizing and Recycling Virtual Volumes and automatically adjust the subgroup assigned. Automatic subgroup adjustment allows CA Vtape to more accurately group data on Backstore Tapes to eliminate the need to Recycle that same data in the future. It also allows any changes to the retention attributes in the parmlib Group Definitions, changes to the Tape Management System rules or manual updates to expiration dates to be automatically applied. This allows the stacking process to change dynamically as your storage management requirements change.

Virtual Volume Stacking Location

When Virtual Volumes are Externalized and Recycled, they are cataloged with a CA Vtape generated name. Technicians or auditors not familiar with CA Vtape are not going to know about the generated name and are not going to be able to track the application data set to the Virtual Volume and then to the Backstore Tape without your help.

When the Interface is activated on a system with CA 1 or CA TLMS installed, Externalization and Recycle will call the interface as each Virtual Volume is Externalized or Recycled. The call will cause CA 1 or CA TLMS to update the ACTVOL field of their volume record with the VOLSER of the Backstore Tape just used.

Having the stacking location recorded in the Tape Management System Virtual Volume record allows anyone using an application data set name query to find the real location of that data set. This information is especially useful in vaulting procedures, auditing vaulting procedures, and disaster recovery scenarios.

By default the update is done with the DUPLEX Backstore Tape VOLSER. If the stacking group assigned to the Virtual Volume does not have duplexing turned on, the update is done with the PRIMARY Backstore Tape VOLSER.

You can use the parmlib Group Definitions attribute TMSVirtualToPhysicalReport to modify the default.

Foreign Tape Processing

A foreign tape is a tape that has not been defined to the Tape Management System. Foreign tapes fall into two categories, VOLSERs that were never defined to the Tape Management System and duplicate VOLSERs.

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

CA Vtape will automatically attempt to intercept any mount for a VOLSER that has been defined to CA Vtape if the unit esoteric or generic coded in the mount request includes the CA Vtape Virtual Devices. If you have a tape with a VOLSER that matches a Virtual Volume, you need to use a unit esoteric or generic that does not include the Virtual Devices, hardcode a device number that does not belong to CA Vtape, or use Foreign Tape Processing to bypass CA Vtape.

When a specific mount request for a CA Vtape VOLSER is detected and the referenced DD statement defines an expiration date that matches the CA Vtape ForeignTapesExpdt attribute value, the Virtual Devices will be marked ineligible in the eligible device list (EDL). The mount request will be ignored by CA Vtape.

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 or CA Vtape will try to intercept the mount request and mount the Virtual Volume instead of the original VOLSER written.

You should define a ForeignTapesExpdt value consistent with that used by the tape management system installed. For example, when mounting a tape that is not controlled by CA 1, you code LABEL=EXPDT=98000 on the DD statement to bypass CA 1 validation. Using the same value for CA Vtape will allow 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. To do this you would set the CA Vtape ForeignTapesExpdt attribute to a value that does not match that of the tape management system and issue the SVT*n* REFRESH=OPTIONS console command to pick up the change. When you code LABEL=EXPDT=98000 in your JCL, you will bypass the tape management system, but not CA Vtape.

Chapter 9: Configuring the Parameter Library (Parmlib)

This chapter documents 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:

General Description (see page 81)
Parmlib Syntax (see page 82)
Parmlib and Symbolic Substitution (see page 83)
Initial Configuration of the Parmlib (see page 85)
Automatic Command Execution (see page 89)

General 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). It contains one or more members that contain CA Vtape parameters known as attributes. 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:

VTPARMS

Parmlib member directory, startup options, and dynamic options.

VTDRIVE

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

VTGROUP

Externalization and DASD buffer definitions.

VTP2POPT

Local TCP/IP definitions for the CA Vtape P2P Option.

VTP2PRMT

Remote TCP/IP definitions for the CA Vtape P2P Option.

VTFILTR

Tape mount intercept filter lists.

VTPOOLS

Volume pools and volser ranges used by CA Vtape.

VTSCMDS

Startup commands.

VTPCMDS

Shutdown commands.

VUSSMNTS

UNIX file system mount points used by USS Backstore.

These members together comprise one complete set of all the attributes used by CA Vtape. Each member contains one or more sections containing information used by CA Vtape 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 documenting the structure of that member, how to customize that member, the attributes contained in the member, and the attributes' valid values.

In general, parmlib members consist of the following:

Blank lines

These are used for legibility and are completely ignored by CA Vtape.

;Comments

Comments always begin with a semicolon. Comments can start in any column. These statements are ignored by CA Vtape and are used for documentation and readability.

<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 imbedded 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.
- There is no line continuation character.
- 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 CA Vtape parameters.
- Values containing blanks or other special characters must be enclosed within single (') or double (“) quote marks. For example, in member VTGROUP, to define the value 3590-1 to the PRIMARY attribute, you will code: Primary='3590-1'.
- Attributes must be coded in their assigned section.
- Multiple parmlib data sets may be concatenated. The first member in the concatenated data sets that contains the section name being searched for will be the one used to load the corresponding attribute values.

Parmlib and Symbolic Substitution

Symbolic substitution is supported 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 and common parmlib members.

Parmlib Attribute Values

CA Vtape supports system symbol substitution within the parmlib members. For example, &SYSNAME resolves to the system name on which CA Vtape 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 attributes to be assigned unique values while sharing parmlib members among different Subsystems.

For example, you define IBM logstreams on LPARS named SYS1 and SYS2 to offload and save the CA Vtape 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 VTPARMS member of the parmlib, find the Logstream attribute and set it to a value of VTAPE.&SYSNAME.LOGSTREAM:

Logstream = VTAPE.&SYSNAME.LOGSTREAM

When CA Vtape is started, the system name will be substituted for the &SYSNAME variable and the appropriate IBM logstream for that LPAR will be accessed.

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

When CA Vtape is started, 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 SVTS PROC. When the S SVTS console command is used to start CA Vtape, and the SVTS parameter is 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

If you wanted to run CA Vtape on two LPARs, SYS1 and SYS2, with different settings for attributes in the Dynamic Options Sections, you could do this from a single parmlib by making the following changes:

1. Edit the VTPARMS member and copy the Dynamic Options Section out to two new members in parmlib named VTDOSYS1 and VTDOSYS2.
2. Find the Parmlib Directory Section in the VTPARMS 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 will be resolved to VTDOSYS1 and initialization will read that member to load the Dynamic Options Section values. When started on SYS2, the DynamicOptions value of VTDO&SYSNAME will be resolved to VTDOSYS2 and initialization will read 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 take place. 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 will be resolved to a value of DOSYSA1. When SVT2 is started on LPAR SYSA, the DynamicOptions attribute will be resolved 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

Complete the following steps to customize the parmlib that was allocated and populated by running the VTA08NON job in the chapter "System Setup."

Detailed explanations of the parmlib attributes and their values can be found in the chapter "The Parameter Library (PARMLIB)." The attribute settings in the sample parmlib members are generally acceptable at any site, but should be reviewed. Where applicable we will direct your attention to a specific attribute to make a recommended change.

Follow these steps:

1. Edit VTPARMS in PREFIX.PARMLIB.

   In the Parmlib Directory Section, uncomment the VolumePoolDefinitions attribute line by deleting the semi-colon at the start of the line.

   Follow the comments in the member to set up the attributes in the Startup Options and Dynamic Options sections.

The following parmlib attributes will need to be updated or reviewed:

Startup Options Section

- DsnameBSDS1 should be set to your BSDS1 data set name.
- DsnameGlobalVcat should be set to your Global VCAT data set name.
- DsnameLocalVcat should be set to your Local VCAT data set name.
- Tasklib is set to Automatic which will dynamically allocate and APF authorize a loadlib data set for use when starting and restarting the Virtual Device Controllers and the Backstore Engine address spaces. If you are executing IBM health checks, this may provoke messages. See Library Mode for this attribute in the chapter "The Paramter Library (PARMLIB)" to prevent the health check messages.
- MessagePrefix should be left at the default value of &SVTS.
- Cache Managment = should be set to Dynamic (recommended)
- CacheDefaultDataClass = should be set to your data class name (Dynamic Cache Management only)
- CacheDefaultStorageClass = should be set to your storage class name (Dynamic Cache Management only)
- ZIIPExploitation should be set to Y if you have a zIIP engine installed or want RMF to track potential zIIP usage.
- LicensedLocalVTP should be set to the number of Virtual Devices you are licensed for use in the CA Vtape Complex.

Dynamic Options Section

- GlobalReserve should be set to Enhanced if you intend to set up more than one CA Vtape Complex in a GRSplex or MIMplex.
- CacheWarningThreshold is set to 85. When 85% of the cache is filled with Virtual Volumes that need to be Externalized, start issuing a warning message. If you would like to start receiving those messages sooner, lower the setting.
- CacheAutoHoldLowThreshold is set to 25.When 25% of cache is filled with Virtual Volumes that need to be Externalized, hold the automated queues and stop Externalizing. If you would like to Externalize more Virtual Volumes before stopping, lower the setting.
- CacheAutoReleaseHighThreshold is set to 65. When 65% of the cache is filled with Virtual Volumes that need to be Externalized, release the automated queues and start Externalizing. If you would like to start Externalizing sooner, lower the setting.
- CacheAutomationSchedule should be reviewed and updated to the time frames when the Backstore Engine is allowed to use tape drives to make copies of the Virtual Volumes in cache and mark those Virtual Volumes for deletion.

- ForeignTapesExpdt should be set to the same value used by your Tape Management System Software.
- CatalogManagedDate should be reviewed to ensure you have the correct value for the Tape Management System Software you have installed.
- MountRejectThreshold is set to CANCEL,10. If the scratch tapes mounted by the Subsystem on a Virtual Device are rejected ten times, the job will be cancelled.
- ScratchVolumesThreshold should be reviewed. Setting it to a high value like 6500 to begin with ensures that you are warned of a pending shortage in plenty of time to address it. It also allows you the time to determine the number of Virtual Volumes you will use and scratch daily so you can calculate a better threshold value.
- TapeManagementSystem should be changed to NONE if you do not have CA 1 or CA TLMS Tape management Systems installed.
- PercentRunOnzIIP should be changed 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.
- RealStorageSafetyThreshold should be reviewed. For execution on a test system with limited memory, a setting of 400 maybe too aggressive. Reducing this to 50 or turning it off with a setting of 0 would be recommended.

2. Edit VTPOOLS.

   Follow the comments in the member to customize the Volume Pool Definitions Section with your customized volume pool names. Update the Volume Pool Sections with your customized pool names and the unique range of Virtual VOLSERs belonging to each pool defined. One pool must be defined.

   Note: If you have a tape management system, the VOLSER range or ranges should first be defined in the tape management system to allow it to track and control the use of the Virtual Volumes just like physical tapes. The range or ranges defined should be reserved solely for the use of CA Vtape.

3. Edit VTDRIVE.

   Follow the comments in the member to complete the Virtual Device range with the primary set of Virtual Devices.

   We recommend that you change the OFFLINE attribute to ONLINE to automatically vary the Virtual Devices online at startup of CA Vtape and split the Virtual Devices between at least two Virtual Control Units. This will cause CA Vtape to start two Virtual Control Unit address spaces, SVTSAS.SVT1V1 and SVTSAS.SVT1V2.

4. Create VTDRALT.

   Copy VTDRIVE into VTDRALT and change the primary set of Virtual Devices to the alternate set. This member will be used 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 VTGROUP.

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

6. Edit VTFILTR.

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

   If Data Class filters are entered, make the corresponding changes in the SMS ACS routines to assign the entered Data Classes to data sets whose tape mounts should be intercepted by CA Vtape.

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

7. Edit VTSCMDS.

   Follow the comments in the member to adjust the Maxdrives setting as needed or add additional console commands.

   The default setting of Maxdrives is 5. This is the maximum number of physical tape drives that the Backstore Engine is allowed to use at one time to copy Virtual Volumes to physical tape.

8. Secure the SUTPARMS member.

   Copy the generated SUTPARMS member from the SVTJCL library to the parmlib. This will insure that this member, which contains the internal configuration data needed by CA Vtape to initialize new control data sets, will not be accidentally overlaid or deleted and will be 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 may prevent CA Vtape from starting up properly and can result in one or more error messages describing the condition along with 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 check the syntax of the parmlib members and produces a report detailing 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 will indicate the parmlib member, section, attribute, and line and column number of the error as well as the maintenance level of the validation program.

The following is a sample parmlib validation error message:

```
+SVTQ0704E Parmlib error:  888
 Tasklib value AUTOMATCI invalid
   Tasklib = Automatic
             >*<<-- See Line.Col(50.13)
Program SVTSPRMS 16.59 09/29/03 SVT1120-QO44319
 PSW@(X'0CC548EC') Offset@(X'003674') Return.Reason(12.704)
 DD(SVTPARMS) Member(VTPARMS) Section<StartupOptions>
```

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

Repeat this process for each set of customized parmlib members.

Automatic Command Execution

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

To enable this feature, you must define a parmlib member name for the StartupCommands and ShutdownCommands attributes in the Parmlib Directory Section in the VTPARMS member of parmlib.

Sample members VTSCMDS and VTPCMDS can be found in HLQ.CCUUPARM.

We recommend you activate the startup commands member and add &SVTS SET HSOPEN=ON and &SVTS D P,M as command entries. The first command ensures that the high-speed open option for Recalls and Recycle is always turned on (you must research whether it can be turned on first). The second command documents the current settings for the parmlib attributes, making it easier to track changes and reverse unwanted changes if needed.

Activate the shutdown commands member and add &SVTS D G, &SVTS D B, and &SVTS D A as command entries. The first command documents if there are any Virtual Volumes that require Externalization and if Externalization is active. The second command will document what Backstore Tapes are in use for Externalization or Recall. The third command documents if any Virtual Devices are in use and which jobs or started tasks are using them.

Since CA Vtape will not shutdown until all its virtual and Backstore Tape activity has stopped, the last two commands are very useful in determining what, if anything, is preventing CA Vtape from shutting down promptly. If CA Vtape is forced down after a stop command is issued, this information documents which Backstore Tapes, jobs, and started tasks were affected and may need corrective action.

Chapter 10: The Parameter Library (PARMLIB)

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.

This section contains the following topics:

Parmlib Attribute Feature and Location (see page 91)
Parmlib Attributes Related by Feature (see page 96)
VTPARMS Parmlib Member (see page 101)
VTDRIVE Parmlib Member (see page 139)
VTGROUP Parmlib Member (see page 142)
VTP2POPT Parmlib Member (see page 159)
VTP2PRMT Parmlib Member (see page 163)
VTFILTR Parmlib Member (see page 166)
VTPOOLS Parmlib Member (see page 171)
VTSCMDS Parmlib Member (see page 174)
VTPCMDS Parmlib Member (see page 175)
VUSSMNTS Parmlib Member (see page 175)

Parmlib Attribute Feature and Location

The following is a table of parmlib attributes and it documents where each parmlib attribute is located and the CA Vtape product features it is related to. Some attributes are used by multiple product features. Some attributes are located in multiple parmlib members.

Observe 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	VTPARMS DynamicOptions
AllowConcurrentVVEReadAccess	VirtualDeviceEngine	VTPARMS StartupOptions

Attribute	Feature	Location
AutomatedSubgroups	BackstoreEngine	VTGROUP GroupSections
BackstoreBlocksize	BackstoreEngine	VTGROUP GroupSections
BackstorePriority	BackstoreEngine	VTGROUP GroupSections
BackstoreRetryCount	BackstoreEngine	VTPARMS DynamicOptions
BackstoreTimeoutValue	BackstoreEngine	VTGROUP GroupSections
BrightStorEncryption	BackstoreEngine	VTGROUP GroupSections
BrightStorEncryptionDC	BackstoreEngine	VTGROUP GroupSections
BypassOfflinePhysicalDevices	BackstoreEngine	VTPARMS StartupOptions
BypassRLLCompression	DASDBufferCompression	VTPARMS DynamicOptions
BypassUCBChecking	VirtualDeviceEngine	VTPARMS StartupOptions
CacheAutoHoldLowThreshold	BackstoreEngine	VTPARMS DynamicOptions
CacheAutomationSchedule	BackstoreEngine	VTPARMS DynamicOptions
CacheAutoReleaseHighThreshold	BackstoreEngine	VTPARMS DynamicOptions
CacheDefaultDataClass	Miscellaneous	VTPARMS StartupOptions
CacheDefaultStorageClass	Miscellaneous	VTPARMS StartupOptions
CacheFreeSpaceThreshold	Miscellaneous	VTPARMS DynamicOptions
CacheManagement	Miscellaneous	VTPARMS StartupOptions
CachePrimary	Miscellaneous	VTPARMS DynamicOptions
CacheResidenceHours	Miscellaneous	VTGROUP GroupSections
CacheSecondary	Miscellaneous	VTPARMS DynamicOptions
CacheWarningThreshold	BackstoreEngine	VTPARMS DynamicOptions
CatalogManagedDate	■ SubgroupAssignment ■ TapeManagement	VTPARMS DynamicOptions
CatalogManagedSubgroup	SubgroupAssignment	VTGROUP GroupSections
ChpidDeviceList	VirtualDeviceEngine	VTDRIVE Virtual Device List
Command | Cmd	Miscellaneous	■ VTPCMDS ShutdownCommands ■ VTSCMDS StartupCommands
ConsoleCommandTimeout	Miscellaneous	VTPARMS DynamicOptions
ConsoleSuffix	PeerToPeer	VTP2PRMT RemoteSections
DatasetFilters	ParmlibDirectory	VTPARMS ParmlibDirectory

Attribute	Feature	Location
DefaultGroup	BackstoreEngine	VTPARMS DynamicOptions
Description	Miscellaneous	VTGROUP GroupSections
DsnameBSDS1	Miscellaneous	VTPARMS StartupOptions
DsnameGlobalVCAT	Miscellaneous	VTPARMS StartupOptions
DsnameLocalVCAT	Miscellaneous	VTPARMS StartupOptions
DsnameSYSTCPD	PeerToPeer	VTP2POPT PeerToPeerOptions
Duplex	BackstoreEngine	VTGROUP GroupSections
DynamicOptions	ParmlibDirectory	VTPARMS ParmlibDirectory
EMCSTerminalPrefix	PeerToPeer	VTP2POPT PeerToPeerOptions
ExchangeMetadataOnly	PeerToPeer	VTGROUP GroupSections
ExcludeDataSets	Filtering	VTFILTR DataSetFilters
ExcludeDataSetsRemote	Filtering	VTFILTR DataSetFilters
Export	BackstoreEngine	VTGROUP GroupSections
ForeignTapesExpdt	Miscellaneous	VTPARMS DynamicOptions
FullMaxdrivesEnforcement	BackstoreEngine	VTPARSMS DynamicOptions
GlobalReserve	Miscellaneous	VTPARMS DynamicOptions
GroupDefinitions	ParmlibDirectory	VTPARMS ParmlibDirectory
Group01-74	FilteringBackstoreEngine	■ VTFILTR FilterSections ■ VTGROUP GroupDefinitions
HardwareCompressionMethod	DASD Buffer Compression	VTPARMS DynamicOptions
HardwareCompressionOption	DASDBufferCompression	■ VTPARMS DynamicOptions ■ VTGROUP GroupSections
IdleTaskTimeout	PeerToPeer	VTP2POPT PeerToPeerOptions
IncludeDataClass	Filtering	VTFILTR DataSetFilters
IncludeDataClassRemote	Filtering	VTFILTR DataSetFilters
IncludeDataSets	Filtering	VTFILTR DataSetFilters
IncludeDataSetsRemote	Filtering	VTFILTR DataSetFilters
IOCpuTimeout	VirtualDeviceEngine	VTPARMS DynamicOptions
IPaddress	PeerToPeer	VTP2PRMT RemoteSections
LicensedLocalVTD	Miscellaneous	VTPARMS StartupOptions

Attribute	Feature	Location
LicensedRemoteVTD	Miscellaneous	VTPARMS StartupOptions
ListenOnIPaddress	PeerToPeer	VTP2POPT PeerToPeerOptions
ListenOnPort	PeerToPeer	VTP2POPT PeerToPeerOptions
LogCSASize	Logger	VTPARMS StartupOptions
LogDataspaceSize	Logger	VTPARMS StartupOptions
LogDetailLevel	Logger	VTPARMS DynamicOptions
LogStream	Logger	VTPARMS DynamicOptions
MaxClients	PeerToPeer	VTP2POPT PeerToPeerOptions
MaximumCompressionCPU	DASDBufferCompression	■ VTPARMS DynamicOptions ■ VTGROUP GroupSections
MB_Threshold	BackstoreEngine	VTGROUP GroupSections
MediumRetention	SubgroupAssignment	VTGROUP GroupSections
MessagePrefix	Miscellaneous	VTPARMS StartupOptions
MIHTimeoutValue	VirtualDeviceEngine	VTPARMS StartupOptions
MinimumCompressionRate	DASDBufferCompression	■ VTPARMS DynamicOptions ■ VTGROUP GroupSections
MountRejectThreshold	VirtualDeviceEngine	VTPARMS DynamicOptions
NeverExpireDate	SubgroupAssignment	VTPARMS DynamicOptions
NeverExpireSubgroup	SubgroupAssignment	VTGROUP GroupSections
Offline	VirtualDeviceEngine	VTDRIVE VirtualDeviceList
OfflineRemote	VirtualDeviceEngine	VTDRIVE VirtualDeviceList
OffsiteBackstoreCopy	BackstoreEngine	VTGROUP GroupSections
Online	VirtualDeviceEngine	VTDRIVE VirtualDeviceList
OnlineRemote	VirtualDeviceEngine	VTDRIVE VirtualDeviceList
PeerToPeerOptions	PeerToPeer	VTPARMS ParmlibDirectory
PeerToPeerRemotes	PeerToPeer	VTPARMS ParmlibDirectory
PercentRunOnZIIP	VirtualDeviceEngine	VTPARMS DynamicOptions
Pool1-8	VirtualDeviceEngine	VTPOOLS PoolSections
Port	PeerToPeer	VTP2PRMT RemoteSections
Primary	BackstoreEngine	VTGROUP GroupSections

Attribute	Feature	Location
ProtocolFamily	PeerToPeer	VTP2POPT PeerToPeerOptions
RealStorageSafetyThreshold	Miscellaneous	VTPARMS DynamicOptions
RecallAttemptsThreshold	BackstoreEngine	VTGROUP GroupSections
RecallNotificationEvent	BackstoreEngine	VTPARMS DynamicOptions
RecallServer	BackstoreEngine	VTPARMS DynamicOptions
RecallServerTimeout	BackstoreEngine	VTPARMS DynamicOptions
RecvTimeOut	PeerToPeer	VTP2PRMT RemoteSections
RemoteSystem	PeerToPeer	VTP2PRMT PeerToPeerRemotes
RemoteSystemPrimary	PeerToPeer	VTGROUP GroupSections
RemoteSystemSecondary	PeerToPeer	VTGROUP GroupSections
RootDirectory	UnixSystemServices	VTPARMS StartupOptions
RoutingCodeCritical	Miscellaneous	VTPARMS StartupOptions
RoutingCodeNormal	Miscellaneous	VTPARMS StartupOptions
ScratchReuseDelay	VirtualDeviceEngine	VTPARMS DynamicOptions
ScratchVolumesThreshold	VirtualDeviceEngine	VTPARMS DynamicOptions
ShortRetention	SubgroupAssignment	VTGROUP GroupSections
ShutdownCommands	ParmlibDirectory	VTPARMS ParmlibDirectory
SpecialRetentionDate	SubgroupAssignment	VTPARMS DynamicOptions
SpecialRetentionSubgroup	SubgroupAssignment	VTGROUP GroupSections
StartupCommands	ParmlibDirectory	VTPARMS ParmlibDirectory
StartupOptions	ParmlibDirectory	VTPARMS ParmlibDirectory
SubAddressSpaceName	Miscellaneous	VTPARMS StartupOptions
TapeManagementSystem	■ SubgroupAssignment ■ TapeManagement	VTPARMS DynamicOptions
Tasklib	Miscellaneous	VTPARMS StartupOptions
TMSVirtualToPhysicalReport	TapeManagement	VTGROUP GroupSections
UnixSystemServices	UnixSystemServices	VTPARMS StartupOptions
USSMountPionts	ParmlibDirectory	VTPARMS ParmlibDirectory
UpdateRMFStatistics	Miscellaneous	VTPARMS StartupOptions
VirtualControlUnit	VirtualDeviceEngine	VTDRIVE VirtualDeviceList

Attribute	Feature	Location
VirtualDeviceList	VirtualDeviceEngine	VTPARMS ParmlibDirectory
VolserRange	VirtualDeviceEngine	VTPOOLS PoolSections
VolumePool	VirtualDeviceEngine	VTGROUP GroupSections
VolumePoolDefinitions	ParmlibDirectory	VTPARMS ParmlibDirectory
zIIPExploitation	VirtualDeviceEngine	VTPARMS StartupOptions

Parmlib Attributes Related by Feature

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

Observe 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
Backstore Engine	AutomatedSubgroups	VTGROUP Group Sections
	BackstoreBlocksize	VTGROUP Group Sections
	BackstoreEncryption	VTGROUP Group Sections
	BackstoreEncryptionDC	VTGROUP Group Sections
	BackstorePriority	VTGROUP Group Sections
	BackstoreRetryCount	VTPARMS Dynamic Options
	BackstoreTimeoutValue	VTGROUP Group Sections
	BypassOfflinePhysicalDevices	VTPARMS Startup Options
	CacheAutoHoldLowThreshold	VTPARMS Dynamic Options
	CacheAutomationSchedule	VTPARMS Dynamic Options
	CacheAutoReleaseHighThreshold	VTPARMS Dynamic Options
	CacheWarningThreshold	VTPARMS Dynamic Options
	DefaultGroup	VTPARMS Dynamic Options

	Duplex	VTGROUP Group Sections
	Export	VTGROUP Group Sections
	FullMaxdrivesEnforcement	VTPARMS Dynamic Options
	Group01-74	VTGROUP Group Definitions
	MB_Threshold	VTGROUP Group Sections
	OffsiteBackstoreCopy	VTGROUP Group Sections
	Primary	VTGROUP Group Sections
	RecallAttemptsThreshold	VTGROUP Group Sections
	RecallNotificationEvent	VTPARMS Dynamic Options
	RecallServer	VTPARMS Dynamic Options
	RecallServerTimeout	VTPARMS Dynamic Options
	RootDirectory	VTPARMS Startup Options
	Triplex	VTGROUP Group Sections
DASD Buffer Compression	BypassRLLCompression	VTPARMS Dynamic Options
	HardwareCompressionMethod	VTPARMS Dynamic Options
	HardwareCompressionOption	■ VTPARMS Dynamic Options ■ VTGROUP Group Sections
	MinimumCompressionRate	■ VTPARMS Dynamic Options ■ VTGROUP Group Sections
	MaximumCompressionCPU	■ VTPARMS Dynamic Options ■ VTGROUP Group Sections
Filtering	ExcludeDataSets	VTFILTR Data Set Filters
	ExcludeDataSetsRemote	VTFILTR Data Set Filters
	Group01-74	VTFILTR Filter Sections
	IncludeDataClass	VTFILTR Data Set Filters
	IncludeDataClassRemote	VTFILTR Data Set Filters
	IncludeDataSets	VTFILTR Data Set Filters
Logger	LogCSASize	VTPARMS Startup Options
	LogDataspaceSize	VTPARMS Startup Options
	LogDetailLevel	VTPARMS Dynamic Options
	LogStream	VTPARMS Dynamic Options

Miscellaneous	Command | Cmd	■ VTPCMDS Shutdown Commands ■ VTSCMDS Startup Commands
	CacheDefaultDataClass	VTPARMS Startup Options
	CacheDefaultStorageClass	VTPARMS Startup Options
	CacheFreeSpaceThreshold	VTPARMS Dynamic Options
	CacheManagement	VTPARMS Startup Options
	CachePrimary	VTPARMS Dynamic Options
	CacheSecondary	VTPARMS Dynamic Options
	ConsoleCommandTimeout	VTPARMS Dynamic Options
	Description	VTGROUP Group Sections
	DsnameBSDS1	VTPARMS Startup Options
	DsnameLocalVCAT	VTPARMS Startup Options
	DsnameGlobalVCAT	VTPARMS Startup Options
	ForeignTapesExpdt	VTPARMS Dynamic Options
	GlobalReserve	VTPARMS Dynamic Options
	LicensedLocalVTD	VTPARMS Startup Options
	LicensedRemoteVTD	VTPARMS Startup Options
	MessagePrefix	VTPARMS Startup Options
	RealStorageSafetyThreshold	VTPARMS Dynamic Options
	RoutingCodeCritical	VTPARMS Startup Options
	RoutingCodeNormal	VTPARMS Startup Options
	SubAddressSpaceName	VTPARMS Startup Options
	Tasklib	VTPARMS Startup Options
	UpdateRMFStatistics	VTPARMS Startup Options
Parmlib Directory	DatasetFilters	VTPARMS Parmlib Directory
	DynamicOptions	VTPARMS Parmlib Directory
	GroupDefinitions	VTPARMS Parmlib Directory
	ShutdownCommands	VTPARMS Parmlib Directory
	StartupCommands	VTPARMS Parmlib Directory
	StartupOptions	VTPARMS Parmlib Directory
	VirtualDeviceList	VTPARMS Parmlib Directory

	VolumePoolDefinitions	VTPARMS Parmlib Directory
Peer To Peer	ConsoleSuffix	VTP2PRMT Remote Sections
	DsnameSYSTCPD	VTP2POPT Peer To Peer Options
	EMCSTerminalPrefix	VTP2POPT Peer To Peer Options
	ExchangeMetadataOnly	VTGROUP Group Sections
	IdleTaskTimeout	VTP2POPT Peer To Peer Options
	IPaddress	VTP2PRMT Remote Sections
	ListenOnIPaddress	VTP2POPT Peer To Peer Options
	ListenOnPort	VTP2POPT Peer To Peer Options
	MaxClients	VTP2POPT Peer To Peer Options
	PeerToPeerOptions	VTPARMS Parmlib Directory
	PeerToPeerRemotes	VTPARMS Parmlib Directory
	Port	VTP2PRMT Remote Sections
	ProtocolFamily	VTP2POPT Peer To Peer Options
	RecvTimeOut	VTP2PRMT Remote Sections
	RemoteSystem	VTP2PRMT Peer To Peer Remotes
	RemoteSystemPrimary	VTGROUP Group Sections
	RemoteSystemSecondary	VTGROUP Group Sections
Subgroup Assignment	CatalogManagedDate	VTPARMS Dynamic Options
	CatalogManagedSubgroup	VTGROUP Group Sections
	MediumRetention	VTGROUP Group Sections
	NeverExpireDate	VTPARMS Dynamic Options
	NeverExpireSubgroup	VTGROUP Group Sections
	SpecialRetentionDate	VTPARMS Dynamic Options
	SpecialRetentionSubgroup	VTGROUP Group Sections
	ShortRetention	VTGROUP Group Sections
	TapeManagementSystem	VTPARMS Dynamic Options
Tape Management	TapeManagementSystem	VTPARMS Dynamic Options
	TMSVirtualToPhysicalReport	VTGROUP Group Sections
UNIIX System Services	DSName	VUSSMNTS Mount Points
	FSName	VUSSMNTS Mount Points

	Parms	VUSSMNTS Mount Points
	RootDirectory	VTPARMS Startup Options
	UnixSystemServices	VTPARMS Startup Options
	USSMountPoints	VTPARMS Parmlib Directory
Virtual Device Engine	AllowBTEncryptionForVVE	VTPARMS Dynamic Options
	AllowConcurrentVVEReadAccess	VTPARMS Startup Options
	BypassUCBChecking	VTPARMS Startup Options
	ChpidDeviceList	VTDRIVE Virtual Device List
	ExchangeMetadataOnly	VTGROUP Group Sections
	IOCpuTimeout	VTPARMS Dynamic Options
	MIHTimeoutValue	VTPARMS Startup Options
	MountRejectThreshold	VTPARMS Dynamic Options
	Offline	VTDRIVE Virtual Device List
	OfflineRemote	VTDRIVE Virtual Device List
	Online	VTDRIVE Virtual Device List
	OnlineRemote	VTDRIVE Virtual Device List
	Pool1-8	VTPOOLS Pool Sections
	PercentRunOnZIIP	VTPARMS Dynamic Options
	RemoteSystemPrimary	VTGROUP Group Sections
	RemoteSystemSecondary	VTGROUP Group Sections
	ScratchReuseDelay	VTPARMS Dynamic Options
	ScratchVolumesThreshold	VTPARMS Dynamic Options
	VirtualControlUnit	VTDRIVE Virtual Device List
	VolserRange	VTPOOLS Pool Sections
	VolumePool	VTGROUP Group Sections
	zIIPExploitation	VTPARMS Startup Options

VTPARMS Parmlib Member

The SVTS JCL procedure for CA Vtape contains a PARMDIR= parameter and an SVTPARMS DD for the parmlib. At startup, SVTS searches the SVTPARMS DD for the parmlib member named by the PARMDIR parameter (the default is VTPARMS). This member must contain an attribute section named ParmlibDirectory. This section documents which parmlib members contain the attribute sections required by CA Vtape.

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

PARMLIB DIRECTORY

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

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

DatasetFilters=

Valid Values:

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

Required: Yes

Description: Parmlib member that contains the Dataset Filters Section.

DynamicOptions=

Valid Values:

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

Required: Yes

Description: Parmlib member that contains the Dynamic Options Section.

GroupDefinitions=

Valid Values:

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

Required: Yes

Description: Parmlib member that contains the Group Definitions Section.

PeerToPeerOptions

Valid Values:

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

Required: No

Description: Parmlib member containing the Peer To Peer Option Section.

Note: This section requires a valid installed license for the CA Vtape P2P performance action.

PeerToPeerRemotes

Valid Values:

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

Required: No

Description: Parmlib member containing the Peer To Peer Remote Section.

Note: This section requires a valid installed license for the CA Vtape P2P performance action.

StartupCommands=

Valid Values:

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

Required: No

Description: Parmlib member that contains the Startup Commands Section.

StartupOptions=

Valid Values:

- VTPARMS (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.

USSMountPoints

Valid Values:

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

Required: Only if UnixSystemServices=Y is specified

Description: Parmlib member containing sections that describe mount points used by USS Backstore.

VirtualDeviceList=

Valid Values:

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

Required: Yes

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

VolumePoolDefinitions

Valid Values:

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

Required: No

Description: Parmlib member containing the Volume Pool Definitions Section.

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.

Observe the following:

- Values in the Valid Values column enclosed in quotes must be entered with quotes, for example, DsnameLocalVCAT='CAVTAPE.SYS1.VCAT'
- Values in the Valid Values column 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. A change to any of these attributes will not take effect until CA Vtape is stopped and started.

AllowConcurrentVVEReadAccess=

This option is available only on JES2 systems and only if CacheManagement=Dynamic is enabled.

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

Valid values:

- NONE
- CADISK

Default: NONE

Feature: Virtual Device Engine

BypassOfflinePhysicalDevices=

Determines whether offline physical devices should be bypassed or considered available during allocation for Externalization and Recall tasks,.

Valid Values:

- Y

  Will bypass (not include) offline physical tape drives.

- N

  Will not bypass (include) offline physical tape drives.

Default: Y

Feature: Backstore Engine

BypassUCBChecking=

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

Valid Values:

- N

  Will not bypass validation.

- Y

  Will bypass validation.

Default: N

Feature: Virtual Device Engine

CacheDefaultDataClass=

Valid Values:

- NONE

  A SMS Data Class is not being defined. Set this value only when CacheManagement=STATIC.

- RESET

  Clears the SMS Data Class shared across subsystems from the Global VCAT. Set this value only when converting all the subsystems from DYNAMIC to STATIC.

- Valid SMS Data Class name

  Set this value with the Data Class to be used by CA Vtape to dynamically allocate Virtual Volumes. A valid Data Class defined with the following attributes is required when CacheManagement=DYNAMIC:

```
Volume Count . . . . . . : 16
Data Set Name Type  . . . : EXTENDED
  If Extended . . . . . . : REQUIRED
  Extended Addressability : NO
  Record Access Bias  . . : USER
Space Constraint Relief . : YES
  Reduce Space Up To (%)  : 50
  Dynamic Volume Count  . : 59
```

  Note: The value set for this attribute is being shared with all other 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 will automatically broadcast the change to all other subsystems.

Default: NONE

Feature: Miscellaneous

CacheDefaultStorageClass=

Specifies the SMS storage class construct name used to dynamically define VSAM linear data sets used as cache buffers for Virtual Volumes or Virtual Control Units.

Multiple CA Vtape subsystems sharing the same Global VCAT automatically share and reference the same SMS construct names. This is necessary to consistently manage the DASD cache space within the storage class.

Valid Values:

- NONE

  A SMS Storage Class is not being defined. Set this value only when CacheManagement=STATIC.

- RESET

  Clears the SMS Storage Class shared across subsystems from the Global VCAT. Set this value only when converting all the subsystems from DYNAMIC to STATIC.

- Valid SMS Storage Class name

  Set this value with the Storage Class to be used by CA Vtape to dynamically allocate Virtual Volumes. Use of a valid Storage Class associated in your Storage Group ACS with the designated CA Vtape Storage Groups is required when CacheManagement=DYNAMIC is specified.

  Note: The value set for this attribute is being shared with all other CA Vtape subsystems from 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 will automatically broadcast the change to all other subsystems.

Default: NONE

Feature: Miscellaneous

CacheManagement=

Determines the type of DASD cache space assigned for new scratch mounts of Virtual Volumes.

Valid Values:

- STATIC

  Is based on DIV I/O services, requires 10 pre-allocated VSAM LDS data sets per Virtual Volume, and can be used with or without SMS.

- DYNAMIC

  Is based on SMS Media Manager I/O services, requires a single dynamically allocated SMS managed VSAM data set per Virtual Volume, and generates less CPU overhead than STATIC.

Default: STATIC

Feature: Virtual Device Engine

DsnameBSDS1=

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

Valid Values:

- JCL

  Indicates that the BSDS1 DD in the 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 SVTS JCL procedure, it must point to the same data set name coded for the attribute. An inconsistency will generate a critical error aborting the subsystem initialization. Commenting out or deleting the BSDS1 DD in the SVTS 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 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 SVTS JCL procedure, it must point to the same data set name coded for the attribute. An inconsistency will generate a critical error aborting the subsystem initialization. Commenting out or deleting the GLOBAL DD in the SVTS 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. The default value of JCK indicates that the VCAT DD in the JCL procedure should be used to allocate the Local VCAT.

Valid Values:

- JCL

  Indicates that the VCAT DD in the 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 will generate a critical error aborting the subsystem initialization. Commenting out or deleting the VCAT DD in the SVTS 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 1 to CAVTAPE.PD02.SVT1VCAT.

Default: JCL

Feature: Miscellaneous

LicensedLocalVTD=

Indicates the number of online local virtual devices that your site is licensed to use. In a CA Vtape complex, SVTS will validate that the number of active online local virtual devices in use does not exceed this value. The default value of -1, prevents SVTS from checking the number of online local devices.

If the value is left at -1, an SVT1IR507I message requesting the value be changed will be issued every eight hours by every CA Vtape Subsystem running with the -1 value.

Valid Values: -1 to 32767

Default: -1

Feature: Miscellaneous

LicensedRemoteVTD=

Indicates the number of online remote Virtual Devices that your site is licensed to use. Will bypass validation. In a CA Vtape complex, SVTS will validate that the number of active online remote Virtual Devices in use does not exceed this value. The value of -1, prevents SVTS from checking the number of online remote devices. The default value of 0 indicates that there are no licensed remote virtual drives.

If the value is set to -1, an SVT1IR507I message requesting the value be changed will be issued every eight hours by every CA Vtape Subsystem running with the -1 value.

Valid Values: -1 to 32767

Default: 0

Feature: Miscellaneous

LogCSASize=

Specifies the size, in kilobytes, of the ECSA area to temporarily store event log records 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 used by the Internal Logger for event log records. A typical loaded environment with LogDetailLevel=1 and the default of 8 MBs will hold 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 issued by CA Vtape.

Note: The default value should not be changed unless there is another software product using the same message prefix value as CA Vtape.

Valid Values:

- &SVTS

  Will be 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 will start with SVT1 as in SVT1R0200E. All messages issued by subsystem SVT2 will start with SVT2 as in SVT2R0200E.

- Any 4 character value or variable

  If four characters are entered, for example SVTS, all messages issued by all CA Vtape subsystems using this parmlib will 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. At shutdown SETIOS commands are issued to set the MIH value to 2 seconds for the now unusable Virtual Devices.

The MIH timeout interval is dependent on environmental conditions like DASD cache volume response times and the number of chained CCW's in an I/O channel program. If DASD cache volume response times are high or applications like DB2 or Syncsort are writing to Virtual Volumes, a higher value for this attribute will be required. We recommend a setting of 30 seconds.

In general, this attribute should not be set to a value higher than 60 seconds without first discussing the change with CA Technical 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, you can set this attribute to a value of -1. When MIHTimoutValue is set to a value of -1, CA Vtape will not set the MIH timeout.

Important! If you set this attribute to a value of -1 and 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 might require an IPL to correct.

Valid Values: -1 to 9999

Default: 15

Feature: Virtual Device Engine

RootDirectory=

This attribute is used by USS Backstore to describe the directory under which to mount file systems. CA Vtape creates subdirectories under the RootDirectory for the CA Vtape Complex and the USS mount points.

Valid Value: A valid USS directory name that the CA Vtape subsystem is authorized to use and has permissions for read/write access.

Default: /u/users/vtape

Feature: UNIX System Services

RoutingCodeCritical=

Determines the route code or codes to be used by CA Vtape 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 to be used by CA Vtape 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 Backstore Engine, Virtual Controller, and Utility address spaces.

Valid Values: A valid JCL procedure library member

Default: SVTSAS

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 runtime modules will be loaded from the STEPLIB DD or a link listed data set and will not be 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 the Backstore Engine or a Virtual Control Unit is restarted, the run-time modules will be loaded from the STEPLIB DD in the SVTSAS JCL procedure or a link-listed loadlib. If the modules loaded do not match the maintenance level of the modules used by the other SVTSAS tasks, an error report will be 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*.LOAD where *prefix* is the CA Vtape data set prefix, *xxxxxx* is the creation date, and *yyyyyy* is the creation time. The temporary loadlib will be dynamically APF-authorized and loaded with the modules found in the SVTS 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 will then be used to start all the CA Vtape started tasks in this subsystem. If the Backstore Engine or 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 shutdown.

  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. The changed modules will not be loaded until the entire subsystem is stopped and started.

- 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 SVTS and SVTSAS JCL procedures. 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.

  The data set name used should start with the CA Vtape DSN prefix to avoid additional security authorizations.

Note: When running in Automatic or Library mode, if an error occurs during start up with allocating, accessing, loading, or dynamically APF-authorizing the loadlib, CA Vtape will issue error messages and automatically change to Steplib mode.

Default: Disabled

Feature: Miscellaneous

UnixSystemServices=

Enables or disables USS Backstore.

Valid Values:

- Y

  Enables USS Backstore.

- N

  Disables USS Backstore.

Default: N

Feature: UNIX System Services

UpdateRMFStatistics=

Determines if CA Vtape will update 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

  Will cause the CMB to be updated.

- N

  Will cause the CMB to not be updated.

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 systems 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 to be made 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. You must 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 don't have a zIIP processor installed; when you run in a z9 system without a zIIP processor installed you can specify this value and PercentRunOnZIIP = 100 to simulate zIIP eligibility for capacity planning purposes. See Operational Considerations to find additional details on how to monitor and perform capacity planning for zIIP processors.

- N

  Indicates that CA Vtape will 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
```

The following is an alphabetic list of the Dynamic Options attributes:

AllowBTEencryptionForVVE=

Determines when Virtual Volumes are eligible for encryption by CA Tape Encryption.

Valid Values:

- N

  Virtual Volumes are not eligible for encryption.

- Y

  Virtual Volumes are eligible for encryption.

- GROUP

  Make Virtual Volumes eligible for encryption based on the associated group definition. Instead of globally applying encryption options to all Virtual Volumes in your environment, you can selectively influence encryption by specific groups. If this parameter is set to GROUPS, CA Vtape will interrogate the group definitions in the VTGROUP member of parmlib to determine the setting of the BrightStorEncryption attribute. If the physical tapes are set for encryption, then CA Vtape will make the Virtual Volume ineligible for encryption. If its value is NONE, the physical tapes will not be encrypted and the Virtual Volumes will be eligible for encryption.

Default: Y

Feature: Virtual Device Engine

BackstoreRetryCount=

Determines the number of Externalization attempts that will be made before a Virtual Volume is dequeued from the subgroup queues without being Externalized.

Valid Values:

- -1

  Indicates that the last value stored in the Local VCAT for this attribute should be used. To determine what the current setting is, issue the SVT*n* Display Groups console command.

- 1 - 32676

  Indicates the number of retries that should be attempted before the Virtual Volume is dequeued. The Externalization Server will wait one minute between attempts.

Note: We recommend a setting of 20, which will allow 20 attempts over a 20-minute period before a Virtual Volume is dequeued. Since the primary reason for an Externalization failure is the Virtual Volume being remounted by an application, a 20-minute delay will allow most applications enough time to complete their processing and release the Virtual Volume. If applications exist that continuously remount a Virtual Volume, then a higher BackstoreRetryCount value is needed.

Default: -1

Feature: Backstore Engine

BypassRLLCompression=

Determines if Run Length Limited (RLL) compression, accessed through the IBM CSRCESRV service, should be bypassed.

Valid Values:

- Y

  Indicates that RLL compression should not be performed (bypassed).

- N

  Indicates that RLL compression should be performed (not bypassed).

Default: Y

Feature: DASD Buffer Compression

CacheAutoHoldLowThreshold=

Determines at what percentage of Cache-In-Use Subgroup Automation should hold automated subgroup queues. Manual subgroup queues are not affected.

If the CacheWarningThreshold or the CacheAutoHighReleaseThreshold attributes are set to zero, this attribute is not active.

This attribute's value must be less than the CacheAutoHighReleaseThreshold value unless both are set to zero.

Note: For more information about Subgroup Automation, see the chapter "Operational Considerations."

Valid Values: 0 to 100

Default: 0

Feature: Backstore Engine

CacheAutomationSchedule=

Determines the time periods when Subgroup Automation is allowed to release the subgroup queues for automated subgroups. Manual subgroup queues are not affected.

If the CacheWarningThreshold or the CacheAutoHighReleaseThreshold attributes are set to zero, this attribute is not active.

Up to four time periods separated by commas can be coded. For example:

```
CacheAutomationSchedule = (00:00-03:00,06:00-09:00,12:00-15:00,18:00-21:00)
```

The time periods coded must not overlap.

Outside of the time periods coded, the subgroup queues for automated subgroups will be held. When a time period ends, any automated subgroup queue that is released or active will be held. The hold will occur within one minute of the ending time period. For example, if a time period ends at 03:00 all automated subgroup queues will be held and any active Externalization tasks for those queues will be gracefully stopped between 03:00 and 03:01.

Note: For more information about Subgroup Automation, see the chapter "Operational Considerations."

Valid Values:

- (time1-time2,time3-time4)
- (time1-time2,time3-time4,time5-time6)
- (time1-time2,time3-time4,time5-time6,time7-time8)

Default: (00:00-23:59)

Feature: Backstore Engine

CacheAutoReleaseHighThreshold=

Determines at what percentage of Cache-In-Use Subgroup Automation should release automated subgroup queues. Manual subgroup queues are not affected.

Valid Values:

- 0

  Indicates that Subgroup Automation is not active.

- 1-100

  Indicates that automated subgroups are eligible for release when the percentage coded is reached or exceeded.

If the CacheWarningThreshold is set to zero, this attribute is not active.

This attribute's value must be greater than the CacheAutoHoldLowThreshold value unless both are set to zero.

Note: For more information about Subgroup Automation, see the chapter “Operational Considerations.”

Default: 0

Feature: Backstore Engine

CacheFreespaceThreshold=

Determines at what percentage of physical allocations in cache CA Vtape will start deleting Externalized Virtual Volumes from cache. A setting of 90 indicates that when the physical allocations in cache reach 90%, Externalized Virtual Volumes should be deleted from cache. CA Vtape will attempt to maintain 10% of the cache free at all times to support scratch mounts and recalls.

Maintaining this free space eliminates the need to free up space during mount processing. By eliminating space management actions from mount processing, the mount can proceed without delay.

Lowering the setting has the following positive effects:

- Increases the amount of free space in cache.
- Reduces the possibility of experiencing momentary cache shortages when multiple jobs are started at the same time requiring scratch mounts or recalls.
- Reduces the possibility of experiencing momentary mount slowdowns while CA Vtape frees needed cache space.

Lowering the setting has the following negative effects:

- Reduces the number of Virtual Volumes that can be kept in cache.
- Tends to increase the amount of DASD needed for the cache.

The setting is typically lowered when you have a number of jobs being submitted at the same time that require large amounts of cache space. For example, if you have multiple backup jobs being submitted at the same time that will write large amounts of data to Virtual Volumes, lowering the setting or increasing the cache size will be necessary to prevent slower mounts and warning messages about momentary cache shortages while CA Vtape is deleting Virtual Volumes from cache.

The 90% setting has been found to be satisfactory at the majority of customer sites.

Observe the following:

- A Virtual Volume deleted from cache must be recalled from a Backstore Tape if mounted again.
- The CacheResidenceHours attribute in the Group Settings can be used to influence which Externalized Virtual Volumes are deleted from cache first.
- The SVT*n* SET THRESHOLD FREESPACE=*nnn* console command can be used to temporarily override the parmlib value.

Valid Values: 0 to 100

Default: 90

Feature: Miscellaneous

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

Section: <DynamicOptions>

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

CacheWarningThreshold=

Determines the percentage of Cache-In-Use at which the SVT*n*D0908I message is issued to warn of a potential cache shortage.

Valid Values:

- -1

  Indicates that the last value stored (0 - 100) in the Local VCAT for this attribute should be used. To determine what the current setting is, issue the SVT*n* Display Groups console command.

- 0

  Indicates the following:

  - The SVT*n*D0908I warning message should not be issued.
  - The CA Vtape subsystem should release all subgroup queues at start-up.
  - Hold and release of individual subgroup queues should not be allowed. The use of the Group parameter on the SVT*n* SET BACKstore,Hold | Release console command will provoke a validation message.
  - Subgroup Automation cannot be activated.

- 1 to 100

  Indicates the following:

  - The SVT*n*D0908I warning message should be issued at or above this percentage of Cache-In-Use. The message will be repeated every minute for as long as the Cache-In-Use remains at or above this percentage.
  - The CA Vtape subsystem should hold all subgroup queues at start-up.
  - Hold and release of individual subgroup queues should be allowed.
  - Subgroup Automation can be activated.

Note: For more information about Subgroup Automation, see the chapter "Operational Considerations."

Default: -1

Feature: Backstore Engine

CatalogManagedDate=

Identifies the date value used to place a tape under catalog management or catalog retention control in the tape management system. This value is used by the Backstore Engine and Recycle to determine which Virtual Volumes will be assigned to the subgroup defined by the group CatalogManagedSubgroup attribute. It is also used by the Backstore Engine and Recycle to place the Backstore Physical Tapes under catalog retention control.

Valid Values:

- -1

  Indicates that the value last stored in the Local VCAT for this attribute should be used. The value stored during installation is 99000.

- A valid date in *yyddd* or '*yyyy/ddd*' format

  Indicates the date value to use. The value used must not match the values coded for the ForeignTapesExpdt, NeverExpireDate, and SpecialRetentionDate attributes.

Default: -1

Features: Subgroup Assignment and Tape Management

ConsoleCommandTimeout=

Controls the number of seconds before console commands timeout and WTOR SVTnD9920I is issued.

Note: We recommend a setting of 60, which will allow 60 seconds for the 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 assigned to a Virtual Volume when the data set cannot be identified based on data class or data set name filter. One example of this is when a Virtual Volume is mounted on Virtual Device by a specific unit allocation, that is:

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

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

Default: GROUP01

Feature: Backstore Engine / Recycle

ForeignTapesExpdt=

Determines the expiration date value that will cause CA Vtape to bypass a mount request for a VOLSER that matches a CA Vtape Virtual Volume, but is not a CA Vtape Virtual Volume. For example, assume 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 will try to intercept the mount. Adding LABEL=EXPDT=*value*, where *valu*e is that of the ForeignTapesExpdt attribute, to the DD statement in the JCL will cause CA Vtape to bypass or ignore the mount request.

Note: Changing the DD statement UNIT parameter to an esoteric or generic that does not include the CA Vtape virtual devices will also cause 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 to use. 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

FullMaxdrivesEnforcement=

Determines if physical tape drives used by Recall tasks should be counted when enforcing the MAXDRIVES setting.

Note: If MAXDRIVES is set to zero and FullMaxdrivesEnforcement is set to Y(es), no Externalization or Recall tasks will be started. FullMaxdrivesEnforcement=Y and MAXDRIVES=0 are the requirements to designate a CA Vtape subsystem as a Failover Backstore Engine Server.

When enforcing the MAXDRIVES setting the Backstore Engine will stop Externalization tasks to free physical tape drives, but will not stop Recall tasks.

Valid Values:

- Y

  Indicates that Recall task physical tape drives should be counted. Use this setting when limiting total physical tape drive usage by the Backstore Engine. Recall physical drive usage will cause active Externalization tasks to stop. Recall physical drive usage itself will not be limited by the MAXDRIVES setting to prevent job delays.

  Note: Recall tasks will always be started without regard to the MAXDRIVES setting to prevent job delays, but Externalization tasks, if active, will be stopped to reduce usage back down to the limit.

- N

  Indicates that Recall task physical tape drives should not be counted. Use this setting when limiting only the physical tape drives used by the Backstore Engine for Externalization tasks. Recall physical drive usage will not cause Externalization tasks to stop.

Default: Y

Feature: Backstore Engine

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 will be 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, will cause contention and a performance penalty between CA Vtape complexes. When one complex issues the generic reserve to update its Global VCAT, other complexes will have to wait to issue the reserve to update their Global VCAT. Even though the different complexes are using different Global VCATs, updates will be serialized across all complexes.

- Enhanced

  Indicates that the Global VCAT name will be used for the reserve RNAME. This will prevent 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 will issue messages and change to compatibility mode.

Default: Compatibility

Feature: Miscellaneous

HardwareCompressionMethod=

Determines which compression methods can be employed when compression is active and compression is to be used for data written onto a Virtual Volume in the DASD buffer.

Valid Values:

- $NULL

  The default value of $NULL 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 instructs CA Vtape to use 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.

  If you have zIIP enabled hardware you can defer the additional CPU requirements of LZ78 by allowing the compression routines to run on the zIIP processor. You can do this by setting additional parmlib attributes (that is, zIIPExploitation=Y and PercentRunOnZIIP=100).

- ALL

  Instructs CA Vtape to employ all of the compression methods mentioned above except $NULL. That is, CMPSC, RLL, and LZ78 are all used. CA Vtape VTA automatically monitors each compression method and chooses the method achieving the highest compression rate for a given Virtual Volume.

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: $NULL

Feature: DASD Buffer Compression

HardwareCompressionOption=

Determines if compression should be performed on data written to Virtual Volumes in the DASD buffer.

Valid Values:

- N

  Indicates that compression should never be performed.

- Y

  Indicates that compression should always be performed.

- JCL

  Indicates that compression should 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, compression will not be performed.

- GROUP

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

Note: For more information about Virtual Volume Compression, see the chapter "Operational Considerations."

Default: N

Feature: DASD Buffer Compression

IOCpuTimeout=

Determines the number of CPU seconds allowed for each I/O before it is terminated by the operating system with abend code S05B. This 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 is recorded by the Logger.

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. This is the recommended setting unless additional log detail is needed to diagnose a problem.

- 2

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

- 3

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

LogDetailLevel=2 and 3 may generate a large volume of records on an LPAR with a very active CA Vtape 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 will not be performed. The Logger will overwrite the data 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 will issue messages and this attribute will be internally changed to a value of NONE.

Note: The only valid Log Stream name is one that was created using the IBM IXCMIAPU utility. You can 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 will be compressed in each 50 MB segment of data 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 should be compressed. This setting should only be used temporarily. If compression should be turned off permanently, use the HardwareCompressionOption attribute.

- 1 to 99

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

- 100

  Indicates compress all the data written.

Note: For more information about Virtual Volume Compression, see the chapter "Operational Considerations."

Default: 100

Feature: DASD Buffer 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 check is disabled. Any achieved percentage compression or no compression will allow 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 "Operational Considerations."

Feature: DASD Buffer Compression

MountRejectThreshold=

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

Valid Values:

- IGNORE

  Indicates that no action will be taken. CA Vtape will continue selecting Virtual VOLSERs from the scratch cell pool until one is accepted by the tape management system or the pool is empty. This could 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 should be issued. The resulting SVT*n*V1803W message asks the operator whether the job should be cancelled or whether the Mount Reject Policy should be ignored.

- Cancel,*nn*,

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

Note: Cancel,10 is the recommended setting.

Default: IGNORE

Feature: Virtual Device Engine

NeverExpireDate=

Identifies the date value used to place a tape under never-expire or permanent retention control in the tape management system.

This value is used by the Backstore Engine and Recycle to determine which Virtual Volumes will be assigned to the subgroup defined by the group NeverExpireSubgroup attribute.

Note: The value used must match the tape management system permanent or never-expire date. The value used must not match the values coded for the ForeignTapesExpdt, CatalogManagedDate, and SpecialRetentionDate attributes.

Valid Values: A valid date in yyddd or 'yyyy/ddd' format

Default: 99365 or '1999/365'

Feature: Subgroup Assignment

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 but CA Vtape will continue to initialize and run without zIIP processing activated.

Valid Values:

- 0

  Specifies that no work is to be directed to the zIIP processor. It is valid to specify PercentRunOnZIIP=0 and zIIPExploitation=Y at the same time. In this case, CA Vtape will issue warning message SVT*n*I0009W but will go on to build and maintain the internal resources required to use the zIIP processor but no work will be 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 that is to be made eligible for the zIIP processor. A value of 100 is recommended to take full advantage of the zIIP processor. You can use WLM to further set priority and eligibility in between other workloads like CA Tape Encryption that are eligible for the zIIP processor.

Default: 0

Feature: Virtual Device Engine

RealStorageSafetyThreshold=

Specifies the percentage which the z/OS available central storage satisfactory threshold should be multiplied by to determine the point at which CA Vtape should defer mount and backstore 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 and for externalization subtasks. As this activity increases, especially in a larger installation with lots of virtual devices being used concurrently, the amount of real storage usage could 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 the CA Vtape does not cause real storage usage to go lower than the safety point.

For instance, the default of 400% implies that CA Vtape will keep a mount in pending status or defer the Backstore of a subgroup queue if the storage that needs to be acquired will reduce the available real storage to within 400% of the safety point. As other tasks that do not control themselves might be requesting real storage at the same time, the default of 400% is a conservative value to delay CA Vtape real storage requests well before z/OS experiences real storage stress. The default value should be lowered if you see CA Vtape delaying mounts on a system where a storage constrain is not occurring.

If the attribute value is set too high, CA Vtape will issue the SVT*n*V5008I message delaying new Virtual Mount requests or the SVT*n*P1708I message delaying new Externalization requests or both and appear to hang. Previous mounts will continue to be processed. Previous Externalization requests will continue to be processed. Only new requests will be delayed. When previous requests complete and free their storage, a new Virtual Mount request or a new Externalization request will be started unless the freed storage is consumed by some other task. Lowering the attribute value and issuing the SVT*n* REFRESH=OPTIONS console command for each CA Vtape subsystem which needs to pick up the change will eliminate the delays.

To disable this feature, specify a value of 0.

Valid Values: 0 to 900

Default: 400

Feature: Miscellaneous

RecallNotificationEvent=

Determines the mode of communication between a Recall Server and its clients.

Valid Values:

- TIMED,*30 to 300*

  Indicates that the Recall Server and its Virtual Control Unit clients will not immediately post each other when a Recall is needed or a Recall completes. The Recall Server will check every *n* seconds for new Recall requests. The Virtual Control Units will check every n seconds to see if a requested Recall has completed. The minimum time interval is 30 seconds to minimize Global VCAT overhead.

  This mode is required if the Recall Server and its clients are not running on the same LPAR and not running in a sysplex.

- ROUTE

  Indicates that the ROUTE console command will be used by the Virtual Control Units to immediately notify the Recall Server of a Recall request and by the Recall Server to notify the requesting Virtual Control Unit when the Recall is complete. This is the recommended configuration, but does require that the Recall Server and all its clients be running in a sysplex.

  When in Route mode, a TIMED,60 mode is automatically activated as a backup. This ensures that Recall requests and completion notifications are received even if a ROUTE command is not properly processed by the operating system.

Default: TIMED,*n*

Feature: Backstore Engine

RecallServer=

Determines whether the Backstore Engine for a CA Vtape subsystem processes its own Recall requests, processes all Recall requests for its CA Vtape Complex, or notifies another subsystem in the complex so it can perform the Recall requests.

Valid Values:

- COMPATIBILITY

  Indicates that this CA Vtape subsystem will only process Recall requests from its own Virtual Control Units. This mode is provided for backward compatibility with Release 2.0 SP05 and prior maintenance-level subsystems.

- SERVER

  Indicates that this CA Vtape subsystem will process the Recall requests from all Virtual Control Units in this CA Vtape Complex.

- CLIENT

  Indicates that this CA Vtape subsystem will pass its Recall requests to the subsystem defined as the Recall Server in this CA Vtape Complex. If the client subsystem has an active Externalization task using the physical tape required for the Recall request, the request will not be passed to the Recall Server. The request will instead be queued to the active Externalization task.

Note: We recommend that you configure as the Recall Server in a CA Vtape Complex the same subsystem that is configured as the Primary Externalization Server. This will minimize physical tape and drive contention for Externalization and Recall requests.

Default: COMPATIBILITY

Feature: Backstore Engine

RecallServerTimeout=

Determines the number of seconds a subsystem will wait before issuing a critical alert message that a Recall request is not being processed. The value should be set high enough to not issue unnecessary alert messages when a Recall request is delayed due to contention for physical tape drives.

Valid Values:

- 0

  Disables the message. The message will never be issued.

- 1 to 1800

  Indicates the number of seconds to wait before issuing the waiting message.

  Note: Using console automation to intercept the SVT*n*IR300I alert message and issue the SVT*n* Display Backstore console command to determine the status of the Recall Server may be desirable.

Feature: Backstore Engine

ScratchReuseDelay=

This attribute Influences the number of days to delay the reuse of a scratched virtual volume. For example, a virtual volume scratched today should not be considered to satisfy a new scratch mount request until (*n*) days have elapsed.

If the attribute value cannot be honored because there are no *n* day old scratches available the subsystem will take 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 found within the pool is used to temporarily override the attribute for a period of one hour.
- An SVT*n*S7023I message is issued describing that the override is in effect for this pool.

Note: For a P2P scratch mount, the attribute setting in the remote server determines the reuse delay. Also, the temporary override of the attribute value is not performed.

Valid Values: 0 to 128

Default:5

Feature: VirtualDeviceEngine

ScratchVolumesThreshold=

Determines whether the 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 will never be 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 will be 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

SpecialRetentionDate=

Indentifies a range of date values used to place a tape under special retention control in the tape management system.

These values are used by the Backstore Engine and Recycle to determine which Virtual Volumes will be assigned to the subgroup defined by the group SpecialRetentionSubgroup attribute.

Note: The value used must match the tape management system special retention dates. The values used must not match the values coded for the ForeignTapesExpdt, CatalogManagedDate, and NeverExpireDate attributes.

Valid Values: A valid date range in 'yyddd-yyddd' or 'yyyy/ddd-yyyddd' format

Default: '99001-99001' or '1999/001-1999/001'

Feature: Subgroup Assignment

TapeManagementSystem=

Determines if the interface to the CA 1 or CA TLMS tape management systems is active. The interface allows CA Vtape to query the tape management system to determine the current expiration date (remaining retention) of a Virtual Volume. This allows Externalization and Recycle to reduce fragmentation on the Backstore Physical Tapes by assigning a Virtual Volume to a subgroup based on its true retention rather than what was specified in the JCL, which wrote the Virtual Volume. The interface also allows CA Vtape to update the Virtual Volume record Container field in the tape management system with the physical tape volser on which the virtual was stacked. The Container field provides a direct link in the tape management system between the application data set on a Virtual Volume and a physical tape for reporting and auditing purposes and for off-site movement.

Valid Values:

- NONE

  Indicates that the interface is not active.

- AUTOMATIC

  Indicates that CA Vtape should query the Subsystem Interface to determine if CA 1 or CA TLMS is installed and activate the interface with the CA tape management system that is found first.

- CA1

  Indicates that CA Vtape should query the Subsystem Interface to determine if CA 1 is installed and, if installed, activate the interface.

- TLMS

  Indicates that CA Vtape should query the Subsystem Interface to determine if CA TLMS is installed and, if installed, activate the interface.

Default: NONE

Feature: Subgroup Assignment and Tape Management

VTDRIVE Parmlib Member

VTDRIVE is a parmlib member containing Virtual Control Unit numbers and device addresses used by CA Vtape to start Virtual Control Units and virtual tape drives. Virtual tape drives are considered local in scope, unique to a subsystem.

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 ChpidDevList attribute is not specified CA Vtape will look for the device assigned to the //CHPID DD in the SVTS PROC and it will use those CHPIDS for all of devices defined by all Virtual Control Units. If the //CHPID DD is not present, CA Vtape will then use the CHPIDs assigned to the DASD device associated with the BSDS1 control data set.

ChpidDeviceList 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 using the ChpidDeviceList attribute. Only the first 8 CHPIDs found when parsing the device list will be used.

Note: 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 SVTS started task, 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.

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

Section: <VirtualDeviceList>

Feature: Virtual Device Engine

Offline=

OfflineRemote=

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.

Offline= is used for defining local drives. OfflineRemote= is used for defining remote drives. Remote Drives require the additional licensing of the CA Vtape P2P Option.

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.

Section: <VirtualDeviceList>

Feature: Virtual Device Engine

Online=

OnlineRemote=

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.

Online= is used for defining local drives. OnlineRemote= is used for defining remote drives. Remote Drives require the additional licensing of the CA Vtape P2P Option.

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.

Section: <VirtualDeviceList>

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

Section: <VirtualDeviceList>

Feature: Virtual Device Engine

VTGROUP Parmlib Member

The VTGROUP parmlib member contains group attribute definitions. These attributes specify characteristics like the esoteric names used to Externalize Virtual Volumes. These options are considered local in scope, unique to the subsystem (unless noted by the attribute description).

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

Since the group definitions are local definitions that are subsystem-specific, the SVT*n* REFRESH=GROUP console command must be issued for each CA Vtape subsystem sharing a common group member that has been modified. If this is not done, then only the CA Vtape subsystem for which the command was issued will pick up the changes.

The SVT*n* REFRESH=GROUP console command should be executed when *there are no active Externalization subtasks for the groups being changed*. Groups with active Externalization subtasks will not be refreshed and the previous attributes will continue in effect. When changes are needed for the attributes in a group, we recommend that you first hold Externalization for the group, wait for the related Externalization subtasks to complete, and then proceed with refreshing the groups.

Important! If a change is made to the Primary, Duplex, Export, ShortRetention, or MediumRetention attributes, CA Vtape will ignore the last-used physical tape. Scratch tapes will be requested for the first Externalization process after the SVTn REFRESH=GROUP console command is executed or the CA Vtape subsystem is restarted.

VTGROUP contains an index section named 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 and their Group Sections which contain your customized settings.

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.

The value coded for each Group attribute can be customized to any name that allows you to easily identify the purpose of the Group. The sample names provided document how each Group will backstore its Virtual Volumes: Primary_Only, Primary_Duplex, and so on. Group Section names like Test_Primary_Not_Encrypted, Prod_Primary_Duplex_Encrypted, and so on, can be used to self document what data is being written to a Group, how it will be backstored, and other important details.

The customized Group Section name must be coded inside less than and greater than signs (<*Customized_Group_Section_Name*>) somewhere 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 defined by a customizable section name 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 defined in the same parmlib member.

Valid Values: A defined Group Section

Section: <GroupDefinitions>

Feature: Backstore Engine / Recycle

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 specifically noted, each attribute is considered local in scope.

AutomatedSubgroups=

Defines which subgroups are to be automated.

None indicates that all the subgroups for groups using this attribute are not automated. The corresponding Externalization Subgroup Queues will not be automatically released and held by Externalization Automation. None is a stand-alone value that cannot be combined with the other valid settings. If None is combined with the other valid settings an SVT*n*QQ412E error message will be issued.

Any combination of S,M,L indicates that the subgroup or subgroups coded are automated for the groups using this attribute. The corresponding Externalization Subgroup Queues will be automatically released and held by Externalization Automation. This attribute determines which of the three subgroups in a group are automated. If none of the subgroups in a group are automated, the attribute should be commented out or deleted.

The S, M, L, Short, Medium, and Long values for this attribute can be specified in any order or combination in a comma-separated list or with separate attribute statements.

The following are examples of single attributes with one or more values:

- AutomatedSubgroups=M
- AutomatedSubgroups=Short,L
- AutomatedSubgroups=Long,M,S

In this example, instead of entering AutomatedSubgroups=Long,M,S as a single attribute statement with multiple values, the same information is entered as separate attribute statements with the following single values:

- AutomatedSubgroups=Long
- AutomatedSubgroups=M
- AutomatedSubgroups=S

Subgroups that are automated are automatically released and held by the Externalization Server based on DASD buffer threshold settings and time schedules determined by the following attributes in the Dynamic Options Section:

- CacheWarningThreshold
- CacheAutoReleaseHighThreshold
- CacheAutoHoldLowThreshold
- CacheAutomationSchedule

Note: For more information about Subgroup Automation, see the chapter “Operational Considerations.”

Valid Values:

- None | N
- Any combination of S, M, L, Short,
- Medium, and Long

Default: None | N

Section: Group Sections

Feature: Backstore Engine

BackstoreBlocksize=

Defines the block size used by the Backstore Engine to Externalize Virtual Volumes queued to this group.

You can specify a different block size value for each group.

If 0K is specified CA Vtape will select a block size based on one of the following output device types:

- 64K for 3480 or 3490 devices (including high density tapes emulating 3490 devices)
- 256K for high density tapes.

Valid Values:

- 0K
- 16K
- 64K
- 265K

Default: 16K

Section: Group Sections

Feature: Backstore Engine

BackstorePriority=

Determines the externalization priority assigned to a group. This priority is used by the Externalization Server to determine which group's Backstore Subgroup Queues should be Externalized first. Zero is the lowest priority. Nine is the highest priority.

Several other factors are considered when starting a subgroup:

- The number of physical tape drives available versus the number of physical tape drives required by the subgroup.
- The number of subtasks required based on the MB_Threshold value versus the amount of data queued for the subgroup.
- The amount of DASD buffer space used by Virtual Volumes in the subgroup.
- The number of physical tape drives required by the subgroup.

Valid Values: 0 to 9

Default: 1

Section: Group Sections

Feature: Backstore Engine

BackstoreTimeoutValue=

Determines the number of seconds that the Backstore Engine will wait for a tape mount before issuing a message to the operator stating that the mount is taking longer than expected.

Valid Values:

- 0

  Indicates the message should never be issued.

- 1 to 9999

  Indicates the number of seconds to wait before issuing the message.

Section: Group Sections

Feature: Backstore Engine

BrightStorEncryption=

Determines which Backstore Tapes are to be written with CA Tape Encryption.

Valid Values:

- NONE

  Do not encrypt the Backstore Tapes.

- PRIMARY

  Encrypt only the Primary Backstore Tape.

- DUPLEX

  Encrypt only the Duplex Backstore Tape.

- BOTH

  Encrypt the Primary and Duplex Backstore Tapes.

Note: When this attribute is set to a value other than NONE, the BrightStorEncryptionDC attribute must specify the SMS DATACLASS defined for CA Tape Encryption services.

Default: NONE

Section: Group Sections

Feature: Backstore Engine

BrightStorEncryptionDC=

Specifies the SMS DATACLASS name enabling encryption services provided by CA Tape Encryption. Review CA Tape Encryption documentation for instruction on how to setup the DATACLASS to enable encryption.

Note: When this attribute is set to a value other than NONE, the BrightStorEncryption attribute must be set to PRIMARY, DUPLEX, or BOTH.

Valid Values:

- NONE
- A valid SMS DATACLASS name

Default: NONE

Section: Group Sections

Feature: Backstore Engine

CacheResidenceHours=

Determines the number of hours that a Virtual Volume will remain resident in DASD cache after externalization. The actual cache expiration date and time is calculated using the Virtual Volume last reference date.

If all of your Virtual Volumes are in the same group or you do not wish to use this feature, leave CacheResidenceHours set to the default value of zero.

Note: Cache utilization may be such that it will be impossible to honor this parameter absolutely. If CA Vtape cannot honor the requested hours, the SVT*n*VQ432I message will be issued with a date and time higher than the current date and time.

Valid Values: 0 to 999

Default: 0

Section: Group Sections

Feature: Miscellaneous

CatalogManagedSubgroup=

Determines the subgroup that catalog managed or catalog control date Virtual Volumes will be assigned to. The catalog managed date value used is determined by the CatalogManagedDate attribute in the Dynamic Options Section.

Valid Values:

- S

  Indicates that catalog managed date Virtual Volumes should be assigned to the Short Subgroup.

- M

  Indicates that catalog managed date Virtual Volumes should be assigned to the Medium Subgroup.

- L

  Indicates that catalog managed date Virtual Volumes should be assigned to the Long Subgroup.

Default: L

Section: Group Sections

Feature: Backstore Engine

Description=

Determines the title line displayed for the group in the ISPF Interface on the Subgroup Display Panel and the Externalization Queue Display Panel under the Group List option. The value is limited to 30 characters.

Valid Values: 'Any 30 characters'

Section: Group Sections

Feature: Miscellaneous

Duplex=

Determines the unit esoteric or generic that will be used to dynamically allocate a non-CA Vtape tape drive for Externalization or Recycle for the duplex output tape. If a duplex tape should not be written, this attribute should be deleted from the Group Section or commented out by typing a semi-colon (;) at the start of the line.

Note: If the unit esoteric or generic contains special characters, it must be enclosed in single quotes. For example, 3590-1 would be coded as Duplex='3590-1'.

Valid Values: A valid unit esoteric or generic

Section: Group Sections

Feature: Backstore Engine / Recycle

ExchangeMetadataOnly=

Determines whether the data is transmitted to the remote system when the Virtual Volume is created. This setting is only in effect for scratch mounts.

Note: The local system's attribute is propagated to the remote. The remote's attribute is ignored.

Valid Values:

- **N** for no :

  Virtual volume data is transmitted and written at both sites during volume creation. Virtual drive IO throughput is throttled by the bandwidth and performance of the network.

- **Y** for yes:

  Only metadata is transmitted during volume creation, minimizing network traffic as the virtual volume is created. Virtual Volume data is never transmitted unless a remote mount request is issued by the remote system. If the local system is not connected, the remote system cannot access the data.

- **D** for defer:

  Like (Y)es, only metadata is transmitted during volume creation, minimizing network traffic as the Virtual Volume is created. However, unlike (Y), the virtual volume is queued for Externalization. When Externalization is run at the remote site, the Virtual Volume data will be transmitted. A remote mount request issued by the remote system will also transmit the Virtual Volume data. If the local system is not connected, the remote system cannot access the data.

Note: The HLQ.CCUUJCL(DEFERXMT) job may be used to trigger the transmission of the data prior to Externalization.

Valid Values:

- Y for yes
- N for no

Default: N

Section: Group Sections

Feature: PeerToPeer

Export=

Determines the unit esoteric or generic that will be used to dynamically allocate a non-CA Vtape tape drive for Externalization for the export output tape. Blank indicates that an Export tape should not be written. If an export tape should not be written, this attribute should be deleted from the Group Section or commented out by typing a semi-colon (;) at the start of the line.

Note: If the unit esoteric or generic contains special characters, it must be enclosed in single quotes. For example, 3590-1 would be coded as Export='3590-1'.

Note: This attribute is mutually exclusive of the Triplex attribute meaning they cannot both be specified for a Group Section.

Valid Values: A valid unit esoteric or generic

Section: Group Sections

Feature: Backstore Engine

HardwareCompressionOption=

Determines if compression should be performed on data written to Virtual Volumes in the DASD buffer 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 should never be performed.

- Y

  Indicates that compression should always be performed.

- JCL

  Indicates that compression should 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 will not be performed.

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

Default: N

Section: Group Sections

Feature: DASD Buffer Compression

MaximumCompressionCPU=

Determines the maximum amount of data in each 50 MB segment of data written to a Virtual Volume that will be compressed. 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.

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

Valid Values:

- 0

  Indicates that no data should be compressed. This setting should only be used temporarily. If compression should be turned off permanently, the HardwareCompressionOption attribute should be used.

- 1 to 99

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

- 100

  Indicates compress all the data.

Note: For more information about Virtual Volume Compression, see the chapter "Operational Considerations."

Default: 100

Section: Group Sections

Feature: DASD Buffer Compression

MB_Threshold=

Determines when additional Externalization subtasks should be started to Externalize the data in the subgroup faster. The amount queued for Externalization in the subgroup queue is divided by the MB_Threshold value and rounded to the next lowest whole number. For example, if 5000 MBs were queued in the subgroup and the threshold is set to the default of 2000 MBs, 5000 divided by 2000 would be 2.5 rounded down to 2. The Backstore Engine will try to start two Externalization subtasks for this subgroup.

Valid Values: 1 to 999999

Default: 2000

Section: Group Sections

Feature: Backstore Engine

MediumRetention=

Determines the upper limit for the number of days of retention that will cause a Virtual Volume to be assigned to the Medium Subgroup. Zero can only be used when ShortRetention is also set to zero. When a number other than zero is coded, it must be greater than the ShortRetention value.

Valid Values:

- 0

  Indicates that Virtual Volumes with normal-dated retention should not be assigned to the Medium Subgroup. Virtual Volumes with never-expire (permanent), catalog control, or special retention dates can still be assigned to the Medium Subgroup. If the Medium Subgroup should not be used, you must set the NeverExpireSubgroup, CatalogManagedSubgroup, and SpecialRetentionSubgroup attributes to a value other than “M”.

- 1 to 32767

  Indicates the number of days of retention for assignment to the Medium Subgroup. If a Virtual Volume has a number of retention days greater than the ShortRetention value and less than or equal to the MediumRetention value, it will be assigned to the Medium Subgroup.

Section: Group Sections

Feature: Subgroup Assignment

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.

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 “Operational Considerations.”

Valid Values: 1 to 100

Section: Group Sections

Feature: DASD Buffer Compression

NeverExpireSubgroup=

Determines the subgroup that never-expire or permanent retention date Virtual Volumes will be assigned to. The never-expire date value used is determined by the NeverExpireDate attribute in the Dynamic Options Section.

Valid Values:

- S

  Indicates that never-expire date Virtual Volumes should be assigned to the Short Subgroup.

- M

  Indicates that never-expire date Virtual Volumes should be assigned to the Medium Subgroup.

- L

  Indicates that never-expire date Virtual Volumes should be assigned to the Long Subgroup.

Default: L

Section: Group Sections

Feature: Backstore Engine

OffsiteBackstoreCopy=

Determines if any copies of the backstore should be excluded from Recall processing. This attribute facilitates sending copies of Virtual Volumes offsite so only the available copy is considered for Recalls.

Valid Values:

- None
- Both
- Primary
- Duplex
- Triplex

Default: None

Section: Group Sections

Feature: Backstore Engine

Primary=

Determines whether a Primary Backstore Tape Copy of the Virtual Volumes assigned to a group will be created and where to write it. This is a required attribute that must be coded in each Group Section.

Specifying TEMPONLY alters the behavior of the Virtual Volume's retention by immediately placing the volume on the free queue. The volume's retention is then controlled by the CacheResidenceHours attribute.

Note: If the unit esoteric or generic contains special characters, it must be enclosed in single quotes. For example, 3590-1 would be coded as Primary='3590-1'.

Valid Values:

- A valid unit esoteric or generic

  A non-CA Vtape unit esoteric or generic will cause the appropriate device type to be dynamically allocated by Externalization and Recycle. A Primary Backstore Tape Copy will be created for the Virtual Volumes.

- CACHONLY

  Indicates that the Virtual Volumes should not be Externalized. A Primary Backstore Tape Copy will not be created for the Virtual Volumes. These Virtual Volumes will reside in cache until they are scratched.

- TEMPONLY

  Indicates that the Virtual Volumes should not be Externalized. A Primary Backstore Tape Copy will not be created for the Virtual Volumes. The Virtual Volumes will reside in cache until their CacheResidenceHours limit is reached and/or space is needed in the DASD cache.

  Important! TEMPONLY volumes can be deleted from cache and the data is never Externalized. If the data should be saved, TEMPONLY should not be usedor another system must Externalize them, for example a P2P Remote Server..

Section: Group Sections

Feature: Backstore Engine

RecallAttemptsThreshold=

Limits the number of times a Recall is attempted from any one backstore copy of a Virtual Volume. Once the threshold is exceeded if another eligible backstore copy is present, Recall will be attempted using that copy. Once all copies have exceeded their threshold, the volume will be mounted as an empty volume. This attribute resets itself for each mount. Specifying or defaulting to zero disables this parameter and Recall always uses the backstore copy after applying the OffsiteBackstoreCopy exclusion rule.

Valid Values: 0 to 15

Default: 0

Section: Group Sections

Feature: Miscellaneous

RemoteSystemPrimary=

Determines which remote CA Vtape Subsystem a local CA Vtape Subsystem should communication with first. This is an optional attribute which is only used when the PeerToPeer Option of CA Vtape has been licensed and activated.

The values coded must correspond to a valid RemoteSystem defined in the Peer To Peer Remotes Section of the VTP2PRMT parmlib member. If a RemoteConsoleSuffix is coded, it must be prefaced with a forward slash (/).

Examples:

```
RemoteSystemPrimary=   SYSA.SVT2          ;  RemoteSystem

RemoteSystemPrimary=   /SAV2              ;  /RemoteConsoleSuffix
```

Note: Coding double quotes with no value between the double quotes or coding $Null indicates that the attribute is not in use.

Valid Values:

- “”
- $Null
- A RemoteSystem value
- The RemoteConsoleSuffix value, prefaced by a forward slash (/), of a RemoteSystem

Default: “”

Section: Group Sections

Feature: Peer To Peer

RemoteSystemSecondary=

Determines which remote CA Vtape Subsystem a local CA Vtape Subsystem should communicate with when the subsystem defined by the RemoteSystemPrimary attribute is not responding. This is an optional attribute which is only used when the PeerToPeer Option of CA Vtape has been licensed and activated.

The values coded must correspond to a valid RemoteSystem or RemoteConsoleSuffix defined in the Peer To Peer Remotes Section of the VTP2PRMT parmlib member. If a RemoteConsoleSuffix is coded, it must be prefaced with a forward slash(/).

Examples:

```
RemoteSystemSecondary=   SYSB.SVT3     ;   RemoteSystem

RemoteSystemSecondary=  /SBV3          ;   /RemoteConsoleSuffix
```

Coding double quotes with no value between the double quotes or coding $Null indicates that the attribute is not in use.

Valid Values:

- “”
- $NULL
- A RemoteSystem value
- The RemoteConsoleSuffix value, prefaced by a forward slash (/), of a Remote System.

Default: “”

Section: Group Sections

Feature: Peer To Peer

ShortRetention=

Determines the upper limit for the number of days of retention that will cause a Virtual Volume to be assigned to the Short Subgroup.

Valid Values:

- 0

  Indicates that Virtual Volumes with normal-dated retention should not be assigned to the Short Subgroup. Virtual Volumes with never-expire (permanent), catalog control, or special retention dates can still be assigned to the Short Subgroup. If the Short Subgroup should not be used, you must set the NeverExpireSubgroup, CatalogManagedSubgroup, and SpecialRetentionSubgroup attributes to a value other than S.

- 1 to 32766

  Indicates the number of days of retention for assignment to the Short Subgroup. If a Virtual Volume has less than or equal to this number of days of retention, it will be assigned to the Short Subgroup.

Section: Group Sections

Feature: Subgroup Assignment

SpecialRetentionSubgroup=

Determines the subgroup that special retention or negative retention date Virtual Volumes will be assigned to. The special retention dates used are determined by the SpecialRetentionDate attribute in the Dynamic Options section.

Valid Values:

- S

  Indicates that special retention or negative retention date Virtual Volumes should be assigned to the Short Subgroup.

- M

  Indicates that special retention or negative retention date Virtual Volumes should be assigned to the Medium Subgroup.

- L

  Indicates that special retention or negative retention date Virtual Volumes should be assigned to the Long Subgroup.

Default: S

Section: Group Sections

Feature: Subgroup Assignment

TMSVirtualToPhysicalReport=

Determines whether the Virtual Volume record in the CA 1 or CA TLMS tape management systems is updated with the duplex or primary tape VOLSER. The field updated is the ACTVOL field. This attribute is active only when the TapeManagementSystem attribute in the Dynamic Options Section is set to a value other than NONE.

Valid Values:

- DUPLEX

  Indicates that the tape management system volume records for Virtual Volumes in this group should be updated with the duplex backstore tape VOLSER. If the group does not have the Duplex attribute specified, then a duplex tape is not being written, the update will be done with the primary backstore tape VOLSER.

- PRIMARY

  Indicates that the tape management system volume records for Virtual Volumes in this group should be updated with the primary backstore tape VOLSER.

Default: DUPLEX

Section: Group Sections

Feature: Tape Management

Triplex=

When USS Backstore is active, this attribute determines if externalization writes a triplex copy of the virtual volume to the USS file system. If a triplex copy shall not be written, this attribute can be commented out of the Group Section by typing a semi-colon (;) at the start of the line.

Note: This attribute is mutually exclusive of the Export= attribute, meaning they cannot both be specified in a Group Section.

Valid Value: USS

Section: Group Sections

Feature: Backstore Engine / UNIX System Services

VolumePool=

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

Virtual Volume pools are defined in the Volume Pool Definitions section of the VTPOOLS member.

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

Default: POOL1

Section: Group Sections

Feature: Miscellaneous

VTP2POPT Parmlib Member

Remote Virtual Tape requires a separate CA P2P license.

VTP2POPT is a parmlib member containing the Peer To Peer Options Section. This section describes attributes used for defining and controlling remote Virtual Tape devices using TCP/IP connections.

Peer To Peer Options Section

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

Observe the following:

- Values in the Valid Values section enclosed in quotes must be entered with quotes. For example: DsnameLocalVCAT='CAVTAPE.SYS1.VCAT'
- Values in the Valid Values section 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 will be used.
- Attributes in this section are only loaded when CA Vtape is started. A change to any of these attributes will not take effect until CA Vtape is stopped and started.

DsnameSYSTCPD=

Defines the name of the profile data set associated TCP/IP subsystem. The profile data set is used to resolve a host name to a TCP/IP address. This profile data set must be assigned to the SYSTCPD DD.

If JCL is specified then CA Vtape uses the data set assigned to the SYSTCPD DD specified in the JCL used to start the subsystem.

If a data set name is specified then the program will dynamically allocate the data set to the SYSTCPD DD.

Valid Values: JCL or a valid data set name

Default: JCL

Section: <PeertoPeerOptions>

Feature: PeerToPeer

EMCSTerminalPrefix=

Defines the prefix for the EMCS console terminal name. The P2P Listener allocates an Extended Management Console (EMCS) to route SVTS commands remotely. EMCS requires an eight character console terminal name which must be unique within the sysplex (that is, the same name cannot be used on different LPARs).

The eight character console name created by CA Vtape consists of:

xxxnyyyy

Where:

xxx

Is the EMCSTerminalPrefix.

n

Is the SVTS subsystem number (that is, 1..8).

yyyy

Is the first four characters of the SYSNAME.

Assign a unique three character string value to EMCSTerminalPrefix, if the default value 'SVT' conflicts with the console naming convention used within your environment.

Valid Values: SVT or any 3 character string

Default: SVT

Section: <PeertoPeerOptions>

Feature: PeerToPeer

IdleTaskTimeOut=

Defines how many minutes one of these subtasks may remain idle before terminating. The P2P Listener uses this attribute to release resources as workload demands change. The P2P Listener is a server which runs in the SVT*x*UT address space. It consists of multiple subtasks which handle remote commands, connection requests, files transfers, and console operations. The P2P Listener dynamically attaches subtasks to handle these functions based on demand. Normally, as activity subsides these subtasks return to an idle wait state.

Valid Values: 1 to 1440

Default: 30

Section: <PeertoPeerOptions>

Feature: PeerToPeer

ListenOnIPaddress=

Defines a TCP/IP address if you want to control the IP address used for receiving or establishing remote connections. This may be useful if you are using VIPA to provide a fall back for connection errors. Omitting this parameter or specifying $NULL will allow the CA Vtape P2P Listener program to accept connections over any network interfaces of the local host.

Valid Values: A valid TCP/IP address

Default: $NULL

Section: <PeertoPeerOptions>

Feature: PeerToPeer

ListenOnPort=

Defines a valid TCP/IP port number over which the CA Vtape P2P Listener program will accept remote connections.

We recommend using an IP port range that uses the SVTS subsystem as the last digit. This will allow you to easily enable multiple CA Vtape subsystems on the same LPAR.

The following example assigns port 11001 for SVT1, 11002 for SVT2, and so on:

```
ListenOnPort = 1100&SVTS(-1:1)
```

Valid Values: 0 to 65535

Default: 0

Section: <PeertoPeerOptions>

Feature: PeerToPeer

MaxClients=

Defines the number of Client Drive subtasks to attach. These Client Drives act as servers to process I/O from a remote systems.

Client Drive subtasks are primarily idle until assigned. When a virtual tape drive on a remote system issues a mount directed to this system a Client Drive is dynamically assigned. When the Virtual Volume on that remote system is dismounted the Client Drive returns to an idle state waiting to be reassigned.

Valid Values: 0 to 512

Default: 0

Section: <PeertoPeerOptions>

Feature: PeerToPeer

ProtocolFamily=

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

Valid Values:

- IPV4

  Indicates to use the IPv4 protocol.

- IPV6

  Indicates to use the IPv6 protocol.

Default: IPV4

Section: <PeertoPeerOptions>

Feature: PeerToPeer

TCPName=

Defines the 1 to 8 character name of the TCP/IP address space that the CA Vtape P2P Listener program will connect to. If not specified the system derives the value from the TCP/IP configuration file as described in IBM's *z/OS Communications Server: IP Configuration Guide*.

Valid Value: A valid TCP/IP address space name

Default: $NULL

Section: <PeertoPeerOptions>

Feature: PeerToPeer

VTP2PRMT Parmlib Member

Note: Remote Virtual Tape requires a separate CA P2P license.

VTP2PRMT is a parmlib member containing the Peer To Peer Remotes section. This section describes attributes used for defining TCP/IP parameters of RemoteSystems.

These attributes specify characteristics for RemoteSystems like the TCP/IP IP address and port number used to communicate with those remote systems. Each remote must have a unique RemoteSystem name, ConsoleSuffix, and IP:Port address combination. The CA Vtape P2P Listener will only accept or establish connection with those systems identified within this list.

While active, the CA Vtape subsystem can dynamically reload the <PeerToPeerRemotes> section attributes using the SVT*n* REFRESH=REMOTES console command.

Since the RemoteSystem definitions are local definitions that are subsystem specific, the SVT*n* REFRESH=REMOTES console command must be issued for each CA Vtape subsystem sharing a common VTP2PRMT member that has been modified. If this is not done, then only the CA Vtape subsystem which issued the command will pick up the changes.

The SVT*n* REFRESH=REMOTES console command should be executed when there are no active remote connections active. Active connections will not be refreshed and the previous attributes will continue in effect until the connection is closed.

Peer To Peer Remotes Section

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

Observe the following:

- Values in the Valid Values section enclosed in quotes must be entered with quotes. For example: DsnameLocalVCAT='CAVTAPE.SYS1.VCAT'
- Values in the Valid Values section 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 will be used.
- Attributes in this list are only loaded when CA Vtape is started. A change to any of these attributes will not take effect until CA Vtape is stopped and started.

RemoteSystem=

Specifies the 1 to 18 character name of a Remote System Section defining the attributes used by the CA Vtape P2P Option for communicating with a remote system. You can define up to as many as 16 RemoteSystems that the subsystem may communicate with.

Valid Values: A valid 1 to 18 character name

Section: <PeerToPeerRemotes>

Feature: PeerToPeer

Remote Section

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

Observe the following:

- Values in the Valid Values enclosed in quotes must be entered with quotes. For example: DsnameLocalVCAT='CAVTAPE.SYS1.VCAT'
- Values in the Valid Values 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 will be used.
- Attributes in this section are only loaded when CA Vtape is started. A change to any of these attributes will not take effect until CA Vtape is stopped and started.

ConsoleSuffix=

Specifies the four character string which is used to route normal SVTS commands to remote SVTS subsystems. A local command such as SVTS HELP can be routed to a RemoteSystem by specifying the console suffix in the command:

SVTS.S*xnn* HELP

The console suffix must be 4 characters which uniquely identifies a remote system.

The operator command SVTS D P, displays the remote console suffix. We suggest making the ConsoleSuffix a combination of the SYSTEMID and the CA Vtape subsystem ID.

For example consider a naming convention such as S*xnn*:

Where:

X

Equals the CA Vtape subsystem id, that is, 1..8.

nn

Equals the system identifier.

For example:

- S7SA might indicate SVT7 running on SYSA
- S101 might indicate SVT1 running on SYS01
- S3T1 might indicate SVT3 running on TEST1

Valid Values: 4 character name

Section: <RemoteSystem>

Feature: PeerToPeer

IPaddress=

Defines the TCP/IP address of the RemoteSystem's P2P Listener. This address is used to establish outbound and validate inbound connections with the RemoteSystem.

Outbound connections are those connections initiated by the local system when it needs to communicate with the RemoteSystem.

Inbound connections are those connection initiated by the Remote system when it needs to communicate with the local system.

Valid Values: 1 to 48 character RemoteSystem TCP/IP address

Section: <RemoteSystem>

Feature: PeerToPeer

Port=

Defines the TCP/IP port number of the RemoteSystem's P2P Listener. This address is used to establish outbound and validate inbound connections with the RemoteSystem.

Valid Values: 1 to 48 character RemoteSystem TCP/IP address

Section: <RemoteSystem>

Feature: PeerToPeer

RecvTimeOut=

Defines the maximum number of minutes to wait for a remote response. If the remote system does not respond to a request within this value then the connection is closed and the transaction is terminated.

This prevents CA Vtape from becoming non-responsive due to a communications failure or non-responsive RemoteSystem.

Valid Values: 1 to 1440

Default: 5

Section: <RemoteSystem>

Feature: PeerToPeer

VTFILTR Parmlib Member

The VTFILTR parmlib member contains Data Set Name Filter include and exclude definitions, as well as SMS Data Class include definitions. These definitions determine what data sets are to be processed by CA Vtape, and what groups will be assigned to those data sets. These options are considered local in scope, unique to the subsystem.

Important! You must 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, the SVT*n* REFRESH=FiLTers console command needs to be issued for each CA Vtape subsystem sharing a common filter member that has been modified. If this is not done, then only the CA Vtape subsystem for which the command was issued will pick up the changes.

VTFILTR contains an index section named Dataset Filters that 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 referenced must be unique and must exist in the same member as the Dataset Filters Section. A section that does not exist will provoke 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= |

ExcludeDataSetsRemote= | ExcludeDSNRemote= | ExcludeDSRemote=

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 or ExcludeDataSetsRemote attributes. Each section must have a unique name.

For example:

```
<DatasetFilters>
ExcludeDatasets=ProdDataSetExcludes
ExcludeDS=TestDataSetExcludes

ExcludeDSRemote=Rmt1.Excludes
ExcludeDSRemote=Rmt2.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

Section: <DatasetFilters>

Feature: Filtering

IncludeDatasets= | IncludeDSN= | IncludeDS= |

IncludeDataSetsRemote= | IncludeDSNRemote= | IncludeDSRemote=

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 or IncludeDataSetsRemote attributes. Each section must have a unique name.

If a data set name matches filters in a non-remote section and a remote section, the non-remote filter will be used and the data set will be assigned to a local group.

For example:

```
<DatasetFilters>
IncludeDatasets=ProdDataSetIncludes
IncludeDS=TestDataSetIncludes

IncludeDSRemote=Rmt1.Includes
IncludeDSRemote=Rmt2.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

Section: <DatasetFilters>

Feature: Filtering

IncludeDataClass= | IncludeDC= |

IncludeDataClassRemote= | IncludeDCRemote=

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 or IncludeDataClassRemote attributes. Each section must have a unique name.

If a data class matches filters in a non-remote section and a remote section, the non-remote filter will be used and the data set will be assigned to a local group.

For example:

```
<DatasetFilters>
IncludeDataClass  =ProdDataClassIncludes
IncludeDC         =TestDataClassIncludes

IncludeDCRemote=Rmt1.Includes
IncludeDCRemote=Rmt2.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

Section: <DatasetFilters>

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 CA Vtape Virtual Volumes. Also determines which group (01-04, 11-14, …, or 71-74) will be 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 wildcarded pattern is not allowed.

Valid Values: A valid SMS data class

Section: <IncludeDataclass>

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 pattern value should be intercepted and written to CA Vtape Virtual Volumes. Also determines which group (01-04, 11-14, …, or 71-74) will be assigned to the Virtual Volumes mounted.

Valid Values:

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

Section: <IncludeDatasets>

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'

Section: <ExcludeDatasets>

Feature: Filtering

VTPOOLS Parmlib Member

VTPOOLS is a parmlib member containing the Volume Pool Definitions section.

Note: Specifying a VTPOOLS member that defines Volume Pooling is required when using control data sets that allow more than 510,800 Virtual Volumes to be defined.

Volume Pool Definitions Section

This section serves as an index to the user-defined pool sections, which are used for defining volume pools and VOLSER ranges for use by CA Vtape. Up to eight volume pools can be defined. POOL1 through POOL7 are user-defined pools used to assign VOLSER ranges. POOL8 is a dynamic pool designed for internal use by the CA Vtape Peer-To-Peer Option.

Consider the following:

- Under JES3 only one volume pool can be defined.
- Specifying a VTPOOLS member that defines Volume Pooling is required when using control data sets that allow more than 510,800 Virtual Volumes to be defined.

When a scratch mount occurs, filters defined in the VTFILTR member assign a group number to the mount. The Group Section assigns a pool name defined in the VTPOOLS member. The Pool Section contains a VOLSER range from which a VOLSER is chosen to satisfy the scratch 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 will dynamically add and delete ranges as appropriate using the latest attributes defined within the VTPOOLS member. The volume pool definitions are global in scope, therefore all CA Vtape subsystems sharing the control files are modified when this command is issued.

The attributes associate a user-defined pool section with a specific pool.

POOL*n*=

This section contains attributes which define volser ranges associated with a given virtual volser pool.

Valid Values: A valid user-defined section within the VTPOOLS members which contains attributes for describing volser ranges associated with the given pool. The *user_defined_pool* name can contain any descriptive name but must not contain the symbols < (less than) or > (greater than).

Section: <VolumePoolDefinitions>

Feature: Volume Pooling

The user-defined pool sections that follow this topic contain the attributes.

VolserRange Section

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

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 VolumePool. SVTS considers the first non-matching character of the volser range as the virtual volume prefix. The first character is always considered a prefix character. The suffix characters must be '0' in the low volser range and '9' in the high range. A minimum of 2 suffix characters are required.

Valid Values: low volser range - high volser range

Section: <VolumePoolDefinitions>

Feature: Volume Pooling

VolumePools Section

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

VolumeSize=

Defines the maximum size of the Virtual Volumes in a Virtual Volume Pool. This attribute can be added to each Virtual Volume Pool. A different size can be defined in each pool.

During customization a default Virtual Volume Size is entered into an ISPF panel and initialized into the Local VCAT. Pools that do not have a VolumeSize= attribute coded use the default size. The default size can be displayed with the SVT1 Display Status console command.

When a scratch mount is performed, the Virtual Volume chosen for the mount is assigned a maximum size. If the VolumeSize= attribute is updated with a different value, active Virtual Volumes in the modified pool continue to use the size they were assigned during their scratch mount. The maximum size does not change until they scratch and are reused.

To use this attribute, all CA Vtape Subsystems in the same CA Vtape Complex must be running with their CacheManagement attributes set to Dynamic.

To define VSAM linear data sets to hold Virtual Volumes larger than 2 GBs, the VSAM extended addressability attribute must be defined in the SMS data class used for these allocations. The Data Class that CA Vtape uses can be displayed with the SVT1 Display Cache console command or by executing an IDCAMS LISTCAT for one of the Virtual Volume LDSs.

Some things to keep in mind about Volume Size is that the size or uncompressed capacity of a Virtual Volume can affect some aspects of CA Vtape performance.

Using a larger value for the Volume Size can allow applications to store more data per Virtual Volume. For applications writing large amounts of tape data, this results in fewer tape mounts allowing them to finish faster. This also results in fewer Virtual Volume DASD cache and Backstore Data Set catalog and tape management entries. This could also result in better Backstore and Recycle performance as fewer tape mounts, opens, and closes are performed per Backstore Tape.

Another aspect to consider is how uncompressed Virtual Volume capacities might affect the number of Virtual Volumes that can concurrently reside in the same amount of DASD cache space. For example, 500GB of DASD cache could probably house 225 to 250 fully utilized 2GB Virtual Volumes. Increasing VolumeSize= to 16GB could reduce the total number of Virtual Volumes housed and cache to 30 fully utilized 16GB Virtual Volumes. Without adding more DASD cache space, Virtual Volumes might need to be Externalized and Recalled more frequently. Finally while a 2GB Virtual Volume might only take a few seconds to RECALL, a 16GB Virtual Volume might take 5 to 6 times longer, delaying the time to mount.

Valid Values: DEFAULT, 400M, 800M, 2G, 4G, 8G, 16G

Section: <VolumePoolDefinitions>

Feature: Volume Pooling

Note: VolumeSize= allows the uncompressed capacity of a Virtual Volume to exceed the Default Virtual Volume Size. Default Virtual Volume Size is the maximum uncompressed capacity supported for Static Cache. You cannot convert Virtual Volumes with uncompressed capacities larger than the Default Virtual Volume Size back to Static Cache. These volumes are lost as a Static Cache Subsystem cannot mount the volumes. The volumes must be carefully analyzed and deleted before attempting to convert back to Static Cache.

VTSCMDS Parmlib Member

The VTSCMDS parmlib member contains operator console commands that are processed by CA Vtape after successful initialization. Any MVS, JES, and CA Vtape console command can be specified in this member. If no commands are needed at startup, $NULL can be coded as the value for the StartupCommands attribute in the Parmlib Directory Section of VTPARMS parmlib member.

To define the desired startup console commands, use the CMD or COMMAND attributes in the <StartupCommands> section of the VTSCMDS member, as shown in the following 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 use of the subsystem symbolic &SVTS in the preceding example will ensure that the SVTS command is processed by the current CA Vtape subsystem being initialized.

VTPCMDS Parmlib Member

The VTPCMDS parmlib member contains operator console commands that are processed by CA Vtape before starting the actual shutdown process. Any MVS, JES, and CA Vtape console command can be specified in this member. If no commands are needed at shutdown, $NULL can be coded as the value for the ShutdownCommands attribute in the Parmlib Directory Section of VTPARMS parmlib member.

To define the desired shutdown console commands, use the CMD or COMMAND attributes under the ShutdownCommands parmlib section part of the VTPCMDS member, 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 use of the subsystem symbolic &SVTS in the preceding example will ensure that the SVTS command is processed by the current CA Vtape subsystem that will be shutdown.

VUSSMNTS Parmlib Member

VUSSMNTS is the parmlib member that defines UNIX file systems dynamically mounted by USS Backstore.

All of the SVTS subsystems of a CA Vtape Complex share the same USS mount points and to facilitate this we recommend using the same VUSSMNTS parmlib member.

Mount Point Directory Sections

A UNIX file system is typically represented in z/OS by Network Attached Storage (NAS), an HFS, or ZFS data set. Read or write access to a UNIX file system requires that it be mounted under a directory path. Section names in the VUSSMNTS member describe the directory name under which to dynamically mount a UNIX file system for use by CA Vtape.

Note: For more information about how the UNIX directory structure is designed for use by USS Backstore, see the section USS File Systems (see page 222).

Section names within the VUSSMNTS member must follow a strict naming convention otherwise they are ignored. The following are the section names in this member:

<USSMountPoints>

This <section> is required although no attributes are currently defined or read for this section. The <USSMountPoints> section is reserved and intended for future use.

In the sample VUSSMNTS member provided we use the <USSMountPoints> section to simply document and describe contents of the VUSSMNTS member itself

<Triplex>

This is a required <section>. The file system defined by the <Triplex> section is mounted under the /RootDirectory/Signature/triplex directory path. This is the UNIX directory used to read or write Triplex Backstore copies of Virtual Volumes. The <Triplex> directory must mount the same file system shared by SVTS subsystems of a CA Vtape Complex.

<Triplex/Group*nn*>

Where *nn* represents a valid Group number.

These <sections> are optional and do not have to be defined. The file systems defined by a <Triplex/Group*nn*> section are mounted under the /RootDirectory/Signature/triplex/group*nn* directory path. This allows CA Vtape to isolate a specific group of Triplex Backstore Virtual Volumes onto separate hardware. This might be desirable for performance reasons or if there was an unusually large amount of Virtual Volume data tied to a specific Group.

Note: <Triplex> or <Triplex/Group*nn*> attributes within the VUSSMNTS member all observe the following:

- Some of the attributes in this list are optional. If commented out or not coded, the documented default will be used.
- Attributes in this list are loaded when CA Vtape is started. It is possible to change the file system mount points without restarting the SVTS subsystems but we recommend taking special precautions when doing so.

  Note: For a detailed description how to accomplish this task, see the section File System Reconfiguration (see page 230).

DSName=

This attribute is required and defines the name of the data set containing the file system to be mounted.

For zFS and HFS file systems this attribute specifies the actual name of the z/OS data set.

Network attached storage (NAS) can also be mounted as file systems under the z/OS UNIX using Network File System (NFS). These are typically hardware DASD devices and therefore do not have or use z/OS naming conventions, however z/OS still requires a unique way to identify these NFS devices. For NFS, DSName can be any name chosen by you to uniquely identify the file system being mounted.

Valid Values: A valid 1 to 44 character data set name

Section: <Triplex> or <Triplex/Group*nn*>

Feature: UnixSystemServices

FSType=

This attribute is required and describes the type of file system to being mounted.

Valid Values: HFS, NFS, ZFS

Default Value: NFS

Section: <Triplex> or <Triplex/Group*nn*>

Feature: UnixSystemServices

Parms=

This is an optional attribute intended primarily for use when FSTYPE=NFS is specified. It describes NFS server, IP and path information necessary to identify and mount the network attached device.

Note: For more information, see the guide *z/OS Network Files Systems Guide and Reference*.

Valid Values: 0 to 256 character string

Default Value: NULL$

Section: <Triplex> or <Triplex/Groupnn>

Feature: UnixSystemServices

Chapter 11: 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:

Renaming the SVTS and SVTSAS PROCs (see page 179)
Customizing the PROCs (see page 180)
Starting the CA Vtape Subsystem (see page 181)
Submitting the Static Cache LDSADDxx Jobs (see page 182)
Verifying the Virtual Devices Are Operational (see page 182)
Verifying CA Vtape is Operational (see page 182)

Renaming the SVTS and SVTSAS PROCs

In an environment where multiple CA Vtape Subsystems will be active on the same LPAR, having multiple PROCs with unique names may make it easier for the operations and technical support personnel to monitor and manage CA Vtape. To support this environment the SVTS and SVTSAS PROCs can be renamed.

There are two cautions that must be observed when renaming these PROCS:

- The PROCs cannot be renamed to a CA Vtape subsystem name (SVT*n* where *n* = 1-8).

  This could cause the tasks to be started under the MASTER subsystem instead of JES resulting in JCL errors or other problems.

- If the SVTSAS PROC is renamed, the SubaddressSpaceName attribute in the Startup Options Section of the VTPARMS parmlib member must be changed to the new name.

When renaming and creating multiple sets of PROCs, a pattern like SVTS1 and SVTS1AS, SVTS2 and SVTS2AS, and so on, allows for easy identification of the CA Vtape Subsystem and its address spaces. In these PROCs the SVTS parameter would be set to their corresponding subsystem name (SVTS=SVT1, SVTS=SVT2, and so on).

If you set the TASKLIB attribute to AUTOMATIC or to a loadlib data set name, the STEPLIB DD in the SVTSAS PROC can be commented out. With the STEPLIB DD commented out, there is no need to have multiple copies of the SVTSAS PROC for different CA Vtape Subsystems. You can start an SVT1 Subsystem, and SVT2 Subsystem, and so on and they can all use the same SVTSAS PROC.

The presence of the last digit of the subsystem name in the PROC name would reinforce the need to use the proper console command prefix or SVTS parm value when issuing commands or running reports to investigating a particular subsystem. When investigating the SVTS2 and SVTS2AS tasks, it would logically follow that console commands should be issued with a command prefix of SVT2 and utility jobs should use a parm of SVTS=SVT2.

When starting or stopping the renamed PROCs, S/P SVTS*n*, where *n* is the last digit of the appropriate subsystem number would suffice. When restarting a Virtual Tape Controller address space or the Backstore Engine, the command prefix would control the restart. SVT1 R CU=1 would restart SVTS1AS.SVT1V1, the Virtual Control Unit address space one for the SVT1 Subsystem. SVT2 R B would restart SVTS2AS.SVT2PT, the Backstore Engine address space for the SVT2 Subsystem.

Customizing the PROCs

To customize the PROCS, follow these steps:

1. Copy the SVTS and SVTSAS 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 will be running on multiple LPARs sharing the same 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 CA Vtape 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 will be "PRODSY,1". If subsystem SVT1 is then started on PRODSYSB, its internal system ID will also be "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.

You can avoid this problem by updating the SYSID=&SYSNAME parameter in the SVTS PROC to a unique value less 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. The default SVTSAS PROC has a region size of 64 MB (REGION=64M). The SVTSAS.SVT*n*PT task (Backstore Engine) may need a higher region size depending on the number of tasks performing backstores and recalls that will be active at the same time.

   If the maximum number of tape drives that CA Vtape will be allowed to use is ten and the group definitions are set up to duplex, then the maximum number of externalization tasks is five. If FullMaxdriveEnforcement is set to Y, then the maximum number of recall tasks is ten. The minimum region size would be five times 4 MB or 20 MB. The maximum region size would be ten times 4 MB or 40 MB. The default region of 64 MB will be sufficient.

   If FullMaxdriveEnforcememt is set to N, then the number of recall tasks will not be limited to the above example of ten. The only limit on the number of recall tasks is the maximum number of physical tape drives available. If there are 32 physical tape drives available, then the maximum region size would be 32 times 4 MB or 128 MB. The default region of 64 MB will not be sufficient whenever a high number of recalls are required.

Starting the CA Vtape Subsystem

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

Many messages are issued at startup to document whether key features are active or inactive. Additional messages are issued to document actions taken, whether critical internal components have been found, and the status of the startup of the main task and its daughter tasks.

CA Vtape consists of one main address space with the default name of SVTS, the Backstore Engine subaddress space with the default name of SVTSAS.SVT1PT, the Utility subaddress space with the default name of SVTSAS.SVT1UT, and up to eight Virtual Device Controller subaddress spaces with the default names of SVTSAS.SVT1V*n* (with *n*=1 to 8). The number of Virtual Device Controllers is determined by what you code in the VTDRIVE parmlib member.

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

Submitting the Static Cache LDSADDxx Jobs

If Static Cache Management was chosen and the LDSDEF*xx* jobs executed successfully, the defined static cache LDSs need to be added to the Global VCAT. Follow these steps:

1. Submit the SUBLDSA member in the PREFIX.SVTJCL data set. It will submit all the LDSADD*xx* jobs that were created during the ISPF Customization Process.
2. Verify that all job steps completed successfully. If any of the jobs failed, take appropriate corrective actions and run the jobs again.

Verifying the 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, execute the operating system command V *xxxx*,ONLINE for each of the missing devices. If you want CA Vtape to automatically vary all the defined Virtual Devices online, review the set up comments in the VTDRIVE parmlib member and adjust the syntax as needed.

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

Verifying CA Vtape is Operational

Multiple CA Vtape console commands will be 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*.

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.
   - IDRC is Default or On.
   - Highspeed Open is On.

- 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 set to the correct parmlib member names.
   - CacheWarningThreshold is set to a non-zero value.
   - CatalogManagedDate is correct.
   - MihTimeoutValue is set to 30.
   - CacheManagement is set to Dynamic.
3. Issue the SVT1 Display Active console command to display the Virtual Devices and the DASD buffer totals.

   The first line of the display contains the total DASD buffer size.
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. Issue the SVT1 Display Groups console command to display the Externalization Subgroup Queues. The DASD buffer totals should change by the amount of space consumed by the Virtual Volume that was just written.

   A line for the group and subgroup used will also be displayed and should match the amount of space used. The group number should match the group number of the filter that caused CA Vtape to intercept the mount request.
6. Issue the SVT1 SET BACKstore=Release,Group=*nn*,SubGroup=*x* console command where *nn* is the group number and *x* is the subgroup that contains the Virtual Volume. The command will override the Backstore Automation setting and release the Subgroup Queue for Externalization.
7. The CA Vtape Subsystem periodically checks the Externalization Subgroup Queues for Virtual Volumes that are queued for Externalization to Backstore Tapes. Issuing the SVT1 CHECK BACKstore console command will cause an immediate check for queued Virtual Volumes. If work is queued, the subsystem will then issue the F SVT1PT,START GROUP=*nn*,SUBGROUP=*x* console command to notify the Backstore Engine to start an Externalization Subtask.

   The same command is available to you except that it is issued as a console command and not a modify command. You would use SVT1 START Group=*nn*,SubGroup=*x*.
8. Issue the SVT1 Display Backstore console command to display the Backstore Engine status. A subtask should be active and a tape mount request in progress for the appropriate group and subgroup.

9. When Backstore has completed processing the Virtual Volume, the Backstore Tape and drive will be released and the Externalization Subtask will be stopped. Issue the SVT1 START RECALL=*vvvvvv* console command where *vvvvvv* is the Virtual Volume VOLSER that was just copied to Backstore Tape. This command will cause the Backstore copy of the Virtual Volume to be copied back into and overlay the Cache copy.
10. CA Vtape finds the Backstore Tape containing the Virtual Volume with a catalog lookup of the name generated by the Backstore process. Using ISPF 3.4, enter the DSN Prefix value from the SVTS Display Status console command followed by VVE.V*. An entry for the Virtual Volume externalized by the Backstore process should be displayed. If you are using Dynamic Cache Management, an entry should also be displayed for the dynamic cache linear data set (LDS). Make sure that these entries are in the user catalog defined for CA Vtape use and not in the master catalog or some other user catalog.
11. Submit a job to read the recalled Virtual Volume and verify its content.
12. 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 named Flags. This is a four byte field displayed in hexadecimal. The bits that make up this field document the status of the Virtual Volume. Things like whether it can be freed from cache, does it need to be Externalized, how many Externalization Copies were made, and so on. 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 were able to intercept a tape mount and write to a Virtual Volume. CA Vtape successfully mounted a Backstore Tape and created a copy of that Virtual Volume and successfully cataloged the Virtual Volume Copy. CA Vtape successfully remounted the Backstore Tape and copied the Virtual Volume back to the cache. You were able to successfully read the Virtual Volume after the Recall.

You were introduced to the ISPF Interface and various console commands that you may need to use in the future. Reading about these and the other console commands is highly recommended.

We also recommend that you print the *Quick Reference Guide* and review its contents. It contains all the console and SVTSUTIL batch commands and groups them by product feature.

There is one last console command that you should issue, SVT1 HELP. This console command lists the syntax of all CA Vtape console commands. Until you are familiar with the console commands or when you cannot find a copy of the *Quick Reference Guide* or the *Administration Guide*, you can use this command to find the console command you are looking for and its syntax.

If you are doing a trial installation then at this point you have installed and configured CA Vtape successfully to perform the trial. The following chapters of this manual document jobs you should run for a production implementation and present additional topics concerning changes to the basic, recommended configuration that might be more appropriate for your environment.

Chapter 12: 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 (see page 187)
High-Speed Open (HSOPEN) Option (see page 190)
Volume Contention (see page 191)
Preserving External Logger Data (see page 192)
Multiple CA Vtape Subsystems on the Same Logical Partition (see page 192)
Copying CA Vtape Data Sets While They Are In Use (see page 193)

Control Data Set Backup

The CA Vtape BSDS should be considered a critical system file. Maintaining a current backup of the BSDS is critical to insuring recoverability if the hardware unexpectedly fails.

The BSDS is typically backed up after batch processing is complete and after Externalizating the Virtual Volumes that were written to during the previous 24 hours. The Tape Management Database and the user catalog containing the CA Vtape Backstore Tape catalog entries should be backed up at this time also.

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

The backups need to be taken close together so that the current Virtual Volume information in the BSDS matches the Tape Management Database information and they both match the cataloged Backstore Tape information. If this information is too far out-of-sync, CA Vtape may not be able to Recall Virtual Volume Copies without manual intervention and your disaster recovery efforts will be much more complicated.

These backups should be sent off-site for disaster recovery. These backups should not be written to CA Vtape Virtual Volumes.

Backing Up the BSDS

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.

You should 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.

You should 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 will 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)
//
```

Recovering 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 DASD buffer is intact, restore the BSDS from the latest backup and recover the Global VCAT as explained in the topic Recovering the Global VCAT.

All CA Vtape Subsystems participating in the CA Vtape Complex sharing the BSDS to be restored must be shut down in order to perform the 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
//
```

Recovering the Global VCAT

If both the BSDS and the Global VCAT are lost but the DASD buffer 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*.

All CA Vtape Subsystems participating in the same CA Vtape Complex sharing the BSDS and Global VCAT to be recovered must be down in order to perform the recovery.

Recovering the Local VCAT

Since the Local VCAT is a work area for transient data and 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 re-create the lost 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. Issue the SVT*n* Display Status console command for the subsystem and check the status of HSOPEN. If the HSOPEN value is not set to the required value, use the SVT*n* SET HSOPEN console command to change the value.

High-Speed Open (HSOPEN) Option

When the High-Speed Open (HSOPEN) option is activated, the starting block ID of each Virtual Volume stacked on a Backstore Tape is saved in the OWNER field of the catalog entry for the stacked Virtual Volume (*PREFIX*.VVE.Vvolser.PRIMARY/DUPLEX) by Externalization and Recycle. Recall, Externalization, and Recycle retrieves the Backstore Tape catalog entry and use the owner field value is to quickly position the tape to the start of the Virtual Volume data set that needs to be read.

When HSOPEN is deactivated, the starting block ID of each Virtual Volume stacked on a Backstore Tape is not saved. When Recall, Externalization, or Recycle mount a Backstore Tape, open processing must forward space through the cartridge to find the required Virtual Volume data set. Significant delays can be seen depending on where the required Virtual Volume data set is located on the Backstore Tape.

The OWNER field in catalog entries is normally not used. If your site uses the OWNER field, you must deactivate the HSOPEN option until you determine if CA Vtape can be exempted from your OWNER field processing.

The value set for the HSOPEN option is saved in the Local VCAT. The value remains in effect across IPLs until it is changed or a new Local VCAT is created. When a new Local VCAT is defined, HSOPEN is automatically turned on. When a new Local VCAT is defined, HSOPEN is automatically turned on.

Because the setting is in the Local VCAT, it is unique to each CA Vtape subsystem. To activate or deactivate HSOPEN, you must issue the SVT*n* SET HSOPEN= console command for each active subsystem that will be performing Externalization or communicating with a Recycle Job.

You can determine the current setting by executing the SVT*n* Display Status console command for each active subsystem.

Note: The SVT*n* SET HSOPEN= console command was added to the sample VTSCMDS (Startup Commands) parmlib member to ensure that the High-Speed Open option is always set to the appropriate value when CA Vtape is started. The default value is ON. If the OWNER field cannot be used at your site, you will need to change this to OFF.

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 needing an unavailable volume will be cancelled.
- When set to WAIT, the job needing an unavailable volume will wait until the volume becomes available.
- When set to WTOR, message IEF235D is issued when the contention is detected and allows an operator to end the wait. As soon as the volume becomes available the job will continue and no operator action is required.

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

If you are running with CacheManagement=Dynamic on a JES2 system, CA Vtape can be configured 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 and is not available with CacheManagement=Static. 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. This means volume contention is less likely to occur except when jobs are submitted at the same time. If message IEF235D does appear, it should only last a few seconds before the jobs proceed on without operator intervention. CA Disk auto-restores (DMSAR) provide their own serialization and message IEF235D will not occur.

Note: For more information, see the section Concurrent Read Access to Virtual Volumes Created by CA Disk (see page 201).

Preserving External Logger Data

The CA Vtape logging facility writes log events into a dataspace owned by CA Vtape. The logged data is small in size 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 (see page 13).

To ensure that the data written out to the log stream is accessible for a longer period of time, without the cost of keeping the data on DASD, 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 of time allows the data to be used like SMF data to look for trends, create historical reports, and diagnose problems.

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. They can be part of the same CA Vtape Complex and share the same Global VCAT/BSDS/DASD buffer or be independent with their own control data sets and DASD buffer.

Unless specifically indicated in the SVTS JCL procedure, subsystem SVT1 will be started. That means that the SVTS JCL procedures, as delivered, will default to starting SVT1. Subsystem SVT1 is not required to be running for other subsystems to start.

Each subsystem performs a COLD start the first time after an IPL and a WARM start after that. The only requirement to run more than one subsystem is to have different LOCAL VCATs and Virtual Drives.

The SVTS= parameter on the EXEC statement of the SVTS 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.

When the subsystems are part of the same CA Vtape Complex, you will get the same results from running RECYCLE or SVTSUTIL batch commands pointing to SVT1 or any of the other subsystems. However, if a subsystem must be active for the utility to run, you must be sure to point to one of the active subsystems.

Commands and messages are prefixed with the subsystem name. If you type SVTS for a command prefix it defaults to SVT1 and SVT1 will execute and respond. If you need to communicate with SVT2, you must use it as the command prefix. The CA Vtape subsystem processing the command will echo the command using the appropriate prefix.

The following are command and message examples:

```
SVTS D B
SVT1X0100I Command Complete

SVT8 START RECALL=105403
SVT8X0100I Command Complete
F SVT8PT,START RECALL=105403
```

Copying CA Vtape Data Sets While They Are In Use

The Transparent Data Migration Facility (TDMF) software allows you to copy your data sets, while they are in use, from one DASD volume to another and switch the volumes. CA Vtape data sets can be copied while in use with TDMF. We recommend that the BSDS be backed up prior to copying the Global VCAT or BSDS data sets and that any copy operations be performed during a period of low CA Vtape mount activity.

Chapter 13: Virtual Device Engine

This chapter describes how to implement and use the Virtual Device Engine.

This section contains the following topics:

Virtual Device Engine Implementation and Use (see page 195)
How Many Virtual Control Units? (see page 196)
Virtual Device Channel Paths (see page 197)
Virtual Volume Compression (see page 200)
Virtual Volume CA Tape Encryption Interface (see page 201)
Concurrent Read Access to Virtual Volumes Created by CA Disk (see page 201)

Virtual Device Engine Implementation and Use

The Virtual Device Engine of a CA Vtape subsystem can consist of up to eight Virtual Control Units, each one running in a separate subaddress space, and each one controlling a subset of the total Virtual Devices defined for CA Vtape. The number of controllers started is controlled by customizing the VTDRIVE member in the CA Vtape parmlib. If the VTDRIVE member is not customized to take advantage of the multiple Virtual Control Units, all Virtual Devices will run under a single subaddress space named SVTSAS.SVT*n*V1. If multiple Virtual Control Units are defined, the subaddress spaces will be named SVTSAS.SVT*n*V*m*, where *n* = the subsystem number and *m* = 1 to 8.

All console commands issued to CA Vtape are issued to the SVTS started task. Subsequent communications between the SVTS and the SVTSAS started tasks in a subsystem are done using system MODIFY (F) commands.

During shutdown, the subsystem will stop the SVTSAS subaddress spaces.

If multiple controller subaddresses are active and one of these subaddress spaces needs to be stopped and restarted, you can use the SVT*n* Restart CU=*n* console command (where *n* = 1-8). This process is completely safe and independent of all other subsystem processes. Virtual Devices under the control of other Virtual Control Units and Backstore Engine physical tape processes are not affected.

When the subsystem detects a RESTART CU command, it does the following tasks:

- Gracefully stops all Virtual Device tasks in progress for the designated Virtual Control Unit.
- Restarts the corresponding SVTSAS subaddress space.
- Resumes the previous Virtual Device tasks in progress.

You should expect to see the following messages when restarting SVTSAS.SVTS1V1:

```
SVT1 RESTART CU=1
SVT1X0101I Command Scheduled in SVTS
SVT1X0100I Command Complete
SVT1A0001I SVT1V1 Stopping
IEF404I SVTSAS - ENDED - TIME=11.19.02
$HASP395 SVTSAS ENDED
$HASP100 SVTSAS ON STCINRDR
$HASP373 SVTSAS STARTED
IEF403I SVTSAS - STARTED - TIME=11.19.03
SVT1A0000I SVT1V1 Ready
```

The SVT*n* Restart CU=*n* console command affects all the Virtual Devices under the Control Unit being restarted. If only one Virtual Device needs to be restarted, use the SVT*n* Restart Unit=*devn* console command.

How Many Virtual Control Units?

The number of Virtual Control Units can be between 1 and 8. The maximum number of Virtual Devices that can be assigned to a Virtual Control Unit is 100.

We recommend that you define two or more Virtual Control Units. This ensures that you can restart portions of the subsystem, effectively guarding against subsystem outages to address a Virtual Device or Virtual Control Unit problem. Multiple Virtual Control Units also reduce enqueue contention. In particular, the SYSZTIOT enqueue issued by dynamic allocation is spread over more than one address space, which reduces the time spent on virtual mounts and dismounts.

CA Vtape manages the startup and shutdown of Virtual Control Unit subaddress spaces, and diagnostic information is gathered for all Virtual Control Unit address spaces when dumps are generated internally or by the SVT*n* DUMP console command.

The VirtualControlUnit attribute in the VTDRIVE member of parmlib defines the number of Virtual Control Units for a subsystem. The Online and Offline (or both) attributes coded immediately following the VirtualControlUnit attribute indicate which Virtual Devices belong to that Control Unit. The following VTDRIVE example defines four Virtual Control Units each controlling four Virtual Devices:

```
<VirtualDeviceList>
 VirtualControlUnit=1
 Online=0570-0573
 VirtualControlUnit=2
 Online=0574-0577
 VirtualControlUnit=3
 Online=0578-057B
 VirtualControlUnit=4
 Online=057C-057F
```

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 start up after an IPL. CHPID assignment is critical since the operating system architecture will not allow an I/O device to be varied online if it does not have a CHPID assigned.

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

Automatic CHPID assignment can be manually overridden by coding either a CHPID DD in the SVTS PROC or specifying the ChpidDeviceList attribute in the Virtual Control Unit section of parmlib.

The CHPID DD can be coded in the following ways:

- //CHPID DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=*vvvvvv*, where *vvvvvv* is a DASD volume whose CHPIDs are to be used.
- //CHPID DD DISP=SHR,DSN=*your.data.set*, where *your.data.set* has been allocated on the DASD volume whose CHPIDs are to be used.
- //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 to be used.

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

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

Once 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 and physical tape activity, stop CA Vtape, change the VirtualDeviceList attribute in the Parmlib Directory Section of the VTPARMS 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. Since these devices have never had CHPIDs assigned to them, CA Vtape can assign CHPIDs automatically or manually to these devices and vary them 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 will 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 BSDS. The SVTS PROC would be copied to create an SVTS1 and an SVTS2 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 SVTS1 Virtual Devices could be varied online. The SVTS2 Virtual Devices would then be varied offline and the other DASD hardware subsystem upgrade would be performed. When that upgrade is complete, the SVTS2 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 dynamically switch 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, the affected jobs have to be canceled, at least one of the assigned CHPIDs has to be brought back online, CA Vtape has to be stopped and started, and the affected jobs have to be resubmitted.

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. The fact that the first CHPID in the list may have been brought back online does not matter. Failover cannot go back to a CHPID previously used or wrap back to the first CHPID in the list when the last CHPID in the list is taken offline. Once CHPID failover has occurred, the failed CHPID can only be reused if CA Vtape is stopped and started.

Virtual Volume Compression

Virtual Volume compression lets you compress data written to Virtual Volumes in the DASD buffer pool. When Externalized to physical tape or Recycled, compressed Virtual Volumes are not decompressed. Compressed volumes occupy less DASD and physical cartridge usage. This results in better DASD buffer and physical cartridge utilization. Increased virtual cartridge capacity also allows a CA Vtape subsystem to manage more user data in the DASD buffer. Compressed data requires fewer I/Os to read or write at the expense of additional CPU overhead needed to perform the actual compression or expansion of that data.

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

- HardwareCompressionOption=
- MinimumCompressionRate=
- MaximumCompressionCPU=

How Virtual Volume Compression Works

When compression is active, CA Vtape will always compress the first 5 or 6 MBs, depending on whether RLL compression is bypassed or not, of each 50 MBs of application data written to a Virtual Volume. If the compression rate falls below the MinimumCompressionRate, CA Vtape will stop compressing for the remaining 45 or 44 MBs of the 50 MBs written. The first 5 or 6 MBs of each 50 MBs of data written to a Virtual Volume is compressed so that CA Vtape can determine which compression algorithm is the best algorithm to use for the data currently being written.

When the MinimumCompressionRate is a low value, a larger percentage of the Virtual Volumes will be compressed and more CPU will be used for compression. When the MinimumCompressionRate is a high value a smaller percentage of the Virtual Volumes will be compressed and less CPU will be used for compression.

Using compression reduces I/O and storage space at the cost of additional 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 controling the percentage of data on which compression is performed.

Specifying a value of 100 would allow all data written to a Virtual Volume to be compressed, which would incur the highest additional CPU cost while producing the maximum amount of compression.

Specifying a value of 50 would only allow 50 percent of the data written to a Virtual Volume to be compressed. On average this would result in approximately 50 percent less compression, but also at 50 percent less CPU cost.

Specifying a value of 0 would result in no compression because no data would be subject to compression.

A low MaximumCompressionCPU value can drop the actual compression rate below the MinimumCompressionRate specified, effectively turning off compression except for the 5 or 6 MBs of test compression for each 50 MBs of data written to a Virtual Volume.

If a zIIP Engine is installed and the zIIPExploitation and PercentRunOnzIIP parmlib attributes are set appropriately, CA Vtape will zIIP enable the compression work. Depending on the availability of the zIIP engine, the amount of compression work performed on your general processors will be reduced lowering the CPU cost of compression.

Virtual Volume CA Tape Encryption Interface

Any tape, virtual or physical, can be encrypted with CA Tape Encryption. If encryption is activated system-wide for all tapes, CA Vtape Virtual Volumes will be encrypted when written by the applications and encrypted again when Externalized. Being encrypted twice does not cause read or write problems for the application or for CA Vtape. Double encryption does mean that additional system resources were used to encrypt data that was already encrypted.

To prevent double encryption, an interface to CA Tape Encryption is available. The interface is activated by the AllowBTEencryptionForVVE attribute in the DynamicOptions Section of the CA Vtape VTPARMS parmlib member.

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

When activated, the interface may prevent Virtual Volumes from being encrypted when they are written by the applications. It will not prevent the Virtual Volumes from being encrypted when they are Externalized.

Concurrent Read Access to Virtual Volumes Created by CA Disk

Most tape media can only be mounted on one device at a time forcing 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 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 with CacheManagement=Dynamic 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 VTPARMS parmlib member.

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

This allows you to write CA Disk ARCHVOLS to CA Vtape Virtual Volumes and to benefit not only from tape utilization efficiencies but also operational performance improvements during disaster recovery and day-to-day storage management processes.

Chapter 14: The Backstore Engine

This chapter describes the Backstore Engine, its parmlib attributes, and how these attributes can be used to manage and automate its actions. The following sections describe these attributes and automation, provide a recommended setup for the Externalization Servers, and other topics related to the Backstore Engine.

This section contains the following topics:

Starting, Stopping, and Restarting the Backstore Engine (see page 203)
Backstore Tape Capacity and Usage (see page 204)
Externalization Server Subgroup Queue Management (see page 205)
Default Group and Subgroup (see page 206)
Subgroup Automation (see page 207)
Overriding and Resetting Subgroup Automation (see page 209)
Configuring the Backstore Engine in Primary and Failover Roles (see page 211)
Backstore Externalization Throughput (see page 212)
Export Tapes (see page 212)
Releasing Tapes from Backstore Processing (see page 213)
Limiting the Number of Tape Drives used by the Backstore Engine (see page 214)
Automatic Recall Source Switching (see page 216)
Allocation Logic - CA MIM or IBM GRS Support (see page 217)
USS Backstore (see page 220)

Starting, Stopping, and Restarting the Backstore Engine

The default name for this engine is SVTSAS. This name can be changed to any name other than SVTSIO and SVT*n* by modifying the SubAddressSpaceName attribute in the Startup Options Section of the VTPARMS parmlib member. The Backstore Engine and the Virtual Device Engine share the same started task procedure, but use step names of SVT*n*PT and SVT*n*V*m* respectively (where *n* = the subsystem number and *m* = 1 to 8).

When the Subsystem is started, the SVTS task is initialized first. The SVTS task then starts the SVTSAS tasks, including the Backstore Engine.

When the Subsystem is stopped, the SVTS task stops the SVTSAS tasks, including the Backstore Engine, and then stops itself.

All console commands issued to CA Vtape are issued to the SVTS started task. Subsequent communications between the SVTS and SVTSAS started tasks by CA Vtape are done using system MODIFY (F) commands.

If the Subsystem detects a Restart Backstore console command, the following actions are taken:

- All Externalization tasks in progress are gracefully stopped, releasing the tapes and tape devices.
- All Recall tasks in progress are gracefully stopped, releasing the tapes and tape devices.
- The Externalization Subgroup Queues are held.
- The SVTSAS.SVT*n*PT address space is stopped and restarted.

If Subgroup Automation is active, no manual intervention is required. If Subgroup Automation is not active, the only manual intervention required is to release the desired groups so Externalization can resume. For the Recalls in progress, the requester is kept in active status waiting for data and, after a few minutes, the internal Missing Interrupt Handler (MIH) support restarts the Recall. As soon as the Backstore Tape is positioned at the same point where it was before, the requester automatically continues its processing.

If you do not want to wait for MIH to restart the pending Recalls, you can issue the SVT*n* MOUNT console command for each Virtual Volume whose Virtual Device is in Mount Pending status.

Note: For more information about this command, see the chapter “Console Commands”.

Important! If the Backstore Engine is cancelled or forced when Externalization has tapes mounted, these tape should be excluded from additional processing by using the SVT*n* SET BACKstore=EXCVOL,VOLume=volser console command. The cancel or force will prevent the last file sequence number used from being saved. If these tapes are remounted, the Backstore Engine will need to read these tapes sequentially to find the last file sequence number. This could cause a significant delay in Externalization as first the PRIMARY tape is mounted and read sequentially, followed by the DUPLEX tape, if duplexing is active.

Backstore Tape Capacity and Usage

CA Vtape can use any mainframe class physical or virtual tape solution for Externalization. Network File Systems, Hierarchical File Systems, or zLinux File Systems can also be used.

Note: CA Vtape automatically excludes its own Virtual Devices from intercepting Externalization tape mount requests.

When creating two copies, a PRIMARY and a DUPLEX Externalization Tape, the copies do not have to be written to the same type or capacity tape.

Many sites find it advantageous to write the PRIMARY Externalization tape to a virtual tape product, 3490 physical tape, or lower density 3590 physical tape and the DUPLEX Externalization tape to a high-density 3590 physical tape for movement off-site.

When high density tape drives are defined for a group (devices running in 3590 mode or 3490 emulated mode), Externalization and Recycle will not span Virtual Volumes across tapes (create multivolume data sets). Instead, when a Virtual Volume does not fit on the current tape, the current tape is closed, a new tape mounted, and the copy is restarted. If a PRIMARY and DUPLEX copy are being written, the restart will occur for both because they are written at the same time; only the copy that ran out of space will use a new tape.

Because PRIMARY and DUPLEX support does not require that the same media type be used for the two copies, the out-of-space condition could occur on each copy at different times for the same Virtual Volume. This would cause the copy operation to be restarted twice for the same Virtual Volumes.

Externalization Server Subgroup Queue Management

CA Vtape has 32 Externalization Groups. Each of these groups is broken into three subgroups. Each of these subgroups represents a unique queue of Virtual Volumes that need to be externalized. CA Vtape tries to efficiently manage the 96 subgroup queues to maximize the amount of data externalized while using the minimum amount of available system resources.

The Externalization Server checks the status of the subgroup queues in the following circumstances:

- A Virtual Volume is dismounted
- The SVT*n*D0908I cache warning threshold message is issued
- An Externalization Subgroup Queue is released
- Five minutes have passed
- The SVTn CHECK BACKstore console command is issued

If multiple Externalization Subgroup Queues have been released, the following factors are used to determine which queue should be processed first:

Drives-In-Use

CA Vtape will not start an Externalization subtask to a subgroup queue if there are not enough available physical drives or if the number of drives in use by Externalization would then exceed the threshold value established by the console command SVT*n* SET MAXDRIVES=*nn*.

Backstore-Priority

The BackstorePriority is a parmlib attribute assigned within the Group Definitions determines which group's Externalization Subgroup queues should be processed first.

Threshold-Priority

This is a dynamically calculated value based on cache utilization and the setting of the MB_Threshold attribute in the Group Definitions. It is used to determine the number of subtasks which will be started for a subgroup queue. Subgroup cache utilization divided by the MB_Threshold, minus the number of currently active subtasks for this queue, determines the Threshold-Priority or the number of additional subtasks required.

For example, if a subgroup queue currently has two active subtasks, still occupies 10000 MBs of cache, and has a MB_Threshold setting of 2000 MBs, its Threshold-Priority would be 3: int((10000/2000)) -2 = 3.

If Threshold Priority is zero, no additional subtasks are required to serve this queue.

Cache-Utilization

This is the amount of DASD buffer space occupied by the Virtual Volumes held in a subgroup queue.

Drives-Required

This value is determined by the Group Definitions in parmlib. The PRIMARY, DUPLEX and EXPORT esoteric attributes determine if a subgroup queue will require one, two, or three tape drives for Externalization.

The Backstore Engine selects the queue with the highest Backstore-Priority, Threshold-Priority, and Cache-Utilization, that has the lowest Drives-Required.

Default Group and Subgroup

The Group number is assigned during filter processing and the Subgroup during Externalization of a Virtual Volume. Under certain circumstances the Group Number is not assigned because a filter match did not occur. These circumstances include:

- A Virtual Device unit number is coded in the JCL with a data set name that does not match a filter.
- When a Virtual Device is allocated with one data set name and written to with another data set name that does not match a filter.
- A job ABEND occurs during or after mount processing and before the data set is opened.

When the Group Number is missing, CA Vtape will access the DefaultGroup attribute in the Dynamic Options Section of the parmlib and assign its value to the problem Virtual Volume. The default setting of this attribute is Group01. The Subgroup will be assigned according to the normal process of evaluating the expiration date of the Virtual Volume.

Recommendations:

- If you are going to customize CA Vtape to use multiple Groups, set the DefaultGroup value to a unique group not used by normal filter processing. This will allow you to monitor whether or not the DefaultGroup is being used and to correct the software or JCL that is bypassing normal filter processing.
- If a unique Group is used for the DefaultGroup value, periodically check the output of a LIST=BACKSTORE Report to see if the DefaultGroup value is being assigned to Virtual Volumes.
- Customize the DefaultGroup value in the Group Definitions Section of the parmlib and release its Backstore Subgroup Queues so its Virtual Volumes can be Externalized.

Subgroup Automation

When Virtual Volumes are created or updated, the space they occupy on DASD cannot be reassigned or used until they have been Externalized by an active Externalization Server. If DASD buffer usage reaches 100%, Virtual Volumes cannot be mounted or recalled until:

- Virtual Volumes are Externalized.
- Virtual Volumes that were previously Externalized and are currently in use are closed and their space freed.
- More space is added to the DASD buffer.

The console command, SVT*n* Display Groups, provides a concise display of cache utilization information.

Note: For more information on the display commands, see the chapter “Console Commands” in the *Administration Guide*.

The ISPF Interface Tape Devices and Virtual Volumes options also display DASD buffer information.

Note: For more information on the ISPF Interface, see the chapter “ISPF Interface” in the *Administration Guide*.

AutomatedSubgroups is a parmlib attribute assigned within the Group Definitions. It determines which subgroup queues within a group are automatically held or released for Externalization based on DASD buffer utilization thresholds. The CacheWarningThreshold and CacheAutoReleaseHighThreshold attributes must both be greater than zero to enable subgroup automation.

The following parmlib attributes can be found in the DynamicOptions Section in the VTPARMS member.

- CacheWarningThreshold is a parmlib attribute that establishes the DASD buffer percentage value where alert messages are issued. These messages are issued once a minute and appear highlighted on the console (WTO DESC=7,11) until DASD buffer utilization subsequently decreases below the following value:
  - SVT*n*D0908I SVTS Cache Shortage - 085% Utilization Reached
  - SVT*n*D0912I SVTS Cache Shortage Relieved

  If the CacheWarningThreshold is zero, Subgroup Automation is disabled and no alert messages are issued. Use the SVT*n* SET BACKstore=Hold|Release console commands to manually hold or release all subgroup queues. Since no alert messages are issued, you should keep the subgroup queues released at all times. This will allow the Externalization Server to externalize Virtual Volumes as soon as they are dismounted and will maximize reusable DASD buffer space at the expense of more frequent physical tape mount requests.

  If the CacheWarningThreshold is greater than zero, Subgroup Automation can be enabled. When the Backstore Engine starts or restarts it places all groups and subgroups on hold status. When Subgroup Automation is not enabled, use the SVT*n* SET BACKstore=Hold|Release,Group=*nn*,SubGroup=*x* console command to hold or release subgroup queues.

- CacheAutoReleaseHighThreshold is a parmlib attribute that establishes the DASD buffer utilization percentage value where the queues for automated subgroups are to be released for processing by the Externalization Server. The following console message is issued when Externalization Server subgroup queues are released:

  ```
  SVTnD0916I Cache at 072% >= 065%, Automated subgroup queues released
  ```

  If the CacheAutoReleaseHighThreshold is set to zero, Subgroup Automation is disabled. The Externalization Server will not dynamically hold or release the Externalization subgroup queues. The SVT*n* SET BACKstore=Hold|Release,Group=*nn*,SubGroup=*x* console commands must be issued to hold or release all subgroup queues or specific subgroup queues as needed.

  When the DASD buffer utilization percentage is calculated, the percentage is rounded up to the next highest whole number. This means that a single DASD buffer LDS that needs to be externalized will cause the percentage to be equal to one. Consequently, setting CacheAutoReleaseHighThreshold=1 means that whenever there is anything to be Externalized, the automated subgroups are eligible for release.

- CacheAutoHoldLowThreshold is a parmlib attribute that establishes the DASD buffer utilization percentage value where automated subgroups are to be held, not processed by the Externalization Server. The following console message is issued when the subgroups are held:

  ```
  SVTnD0920I Cache at 020% <= 020%, Automated subgroup queues held
  ```

  Note: After the automated subgroups are released, as each subgroup queue is emptied (that is, the number of Virtual Volumes waiting for externalization reaches zero), the subgroup queue is automatically held. This policy helps minimize the number of Backstore mounts and drive allocations in scenarios where a subgroup queue is quickly externalized but the DASD buffer utilization does not reach the CacheAutoHoldLowThreshold.

- CacheAutomationSchedule is a parmlib attribute that defines up to four time periods in a 24-hour period when the Externalization Server is allowed to release automated subgroups. Outside of these time periods the Externalization Server will place all automated subgroup queues in hold status. The following console message is issued when automated subgroup queues are held:

  ```
  SVTnD0924I Cache at 054% Automated subgroup queues held due to automation schedule
  ```

  CA Vtape checks DASD buffer utilization once a minute.

Overriding and Resetting Subgroup Automation

When required, you can override Subgroup Automation by using the SVT*n* SET BACKstore=-Release | -Hold,Group=*nn* console commands. The minus sign turns automation off for the subgroups targeted by the command and then releases or holds all manual subgroups targeted by the command.

For example, if SVT*n* SET BACK=-R,G=31 is issued and subgroups Short and Medium for group 31 are automated, the command would turn off automation for Short and Medium and then release Short, Medium, and Long. The command first made Short and Medium manual groups by turning off their automation and then released all the manual subgroups for group 31 which included Long.

After being deactivated, you can restore Subgroup Automation by issuing the SVT*n* SET BACKstore=reseT,Group=*nn* console command. The command will access the parmlib to determine the defined settings for the subgroups targeted by the command and return the affected subgroups to these settings. This may or may not cause the status of the subgroup queues to change.

The status of an Externalization Subgroup Queue is changed by Subgroup Automation when the Cache In-Use percentage is equal to or greater than the CacheAutoReleaseHighThreshold or equal to or less than the CacheAutoHoldLowThreshold. If Subgroup Automation is reset when the Cache In-Use percentage is between the CacheAutoReleaseHighThreshold and the CacheAutoHoldLowThreshold, the status of the Externalization Subgroup Queues will not be changed. The queues are left in whatever status the last override command put them.

For example, CacheAutoReleaseHighThreshold is set to 50%, CacheAutoHoldLowThreshold is set to 20%, and the status of all the automated Externalization Subgroup Queues is HOLD when the SVT*n* Display Group console command is issued. This is their status because Cache In-Use has not exceeded the release setting. The SVT*n* SET BACKstore=-Release,Group=* console command is issued to override Subgroup Automation and release all the now manual Externalization Subgroup Queues. Externalization lowers the Cache In-Use percentage to 30% and the SVT*n* SET BACKstore=reseT,Group=* console command is issued to reactivate Subgroup Automation. The Externalization Subgroup Queues will remain in RLSE or ACT(n) status when the SVT*n* Display Group console command is issued. The queues will not be reset to HOLD status because the Cache In-Use percentage is not equal to or less than the CacheAutoHoldLowThreshold value.

In this situation, if the status of the queues should be HOLD, issue the SVT*n* SET BACKstore=Hold,Group=* console command to return the automated queues to manual control and hold them. Then issue the RESET console command to restore Subgroup Automation.

If overriding and resetting Subgroup Automation becomes a regular event that must be automated, then you should set the queues to the desired status before the RESET. For example, an hour before the off-site movement of physical tapes, the queues may be released to copy as many Virtual Volumes to physical tape as possible. The commands to automate and the order to automate them would be as follows:

1. SVT*n* SET BACK=-R,G=*, to first deactivate Subgroup Automation and release the now manual queues.
2. SVT*n* SET BACK=H,G=*, to return the queues to HOLD status.
3. SVT*n* SET BACK=EXCLVOL,G=*,T=D, to release the DUPLEX physical tapes for off-site movement.
4. Create the off-site movement pull list.
5. SVT*n* SET BACK=T,G=*, to restore Subgroup Automation.

Note: For more information about the syntax and options of these commands, see the chapter “Console Commands” in the *Administration Guide*.

Configuring the Backstore Engine in Primary and Failover Roles

Based on the values set for some parmlib attributes and settings, a Backstore Engine can be defined as a Primary Server for all Externalization and Recall activities or a Failover Server. We recommend this configuration to minimize the number of tapes, and physical tape drives being used.

The Primary Server will take care of all Externalization and Recall tasks for a CA Vtape Complex installed on one or more LPARs. The Failover Servers will be dormant, waiting to take over this work if needed. With this configuration, you can centrally manage the Externalization and Recall processes from just one LPAR and CA Vtape subsystem without having to track tasks, alerts, commands, and messages from multiple LPARs and subsystems.

The configuration of Primary and Failover Servers in a CA Vtape Complex is very simple. All the CA Vtape Subsystems in the same CA Vtape Complex should share the same Group Definitions and the same Externalization and Recall related attributes in the <DynamicOptions> section of parmlib. All subsystems should also have FullMaxdrivesEnforcement set to a value of Y. These parmlib attributes are:

- CacheAutoHoldLowThreshold
- CacheAutomationSchedule
- CacheAutoReleaseHighThreshold
- CacheWarningThreshold
- FullMaxdrivesEnforcement (must be set to Y)
- RecallNotificationEvent
- RecallServer (must be set to SERVER)
- RecallServerTimeout

Setting these parmlib attributes to the same values will ensure that each Backstore Engine reacts in the exact same way once MAXDRIVES is set to a value greater than zero.

A Backstore Engine is the Primary Server when its MAXDRIVES is set to a number greater than zero. A Backstore Engine is a Failover Server when its MAXDRIVES is set to zero. To accomplish this the SVT*n* SET MAXDRIVES=0 console command is issued for the CA Vtape Subsystems that are the Failover Servers and SVT*n* SET MAXDRIVES=*nn* where *nn* is a value greater than zero for the CA Vtape Subsystem that is the Primary Server.

The Startup Commands Section and the Shutdown Commands Section in the CA Vtape parmlib or your own automation product can be used to automate the issuing of these commands for the Primary and Failover Servers.

When a problem occurs that prevents the Primary Server from processing Recall or Externalization requests, issuing the SVT*n* SET MAXDRIVES=*nn* console command for one of the Failover Servers will make it the Primary Server. If the original Primary Server should not perform any physical tape mounts until the problem is resolved, its MAXDRIVES setting should be changed to zero.

Backstore Externalization Throughput

Externalization throughput and the number of tape cartridges used depend on the output block size. The actual block size is selected by setting and dynamically refreshing the VTGROUP parmlib attribute BackstoreBlocksize. The recommended value for this attribute is 0K specifying that CA Vtape must select the appropriate block size based on the output device type.

Note: If high density tapes (IBM 3590 or STK 9840 or equivalent) are being used in 3490 emulation, set this attribute to 256K. If 0K is used, CA Vtape will select 64KB as the block size which will be inefficient for these tape devices.

If the default block size value of 16K is being used, Externalization on high density tapes can achieve around a 6 MBs per second end-to-end transfer rate when only Primary is defined. End-to-end transfer rate includes the time needed for label processing, cataloging the data set, and waiting for the tape management system response. Externalization with Primary and Duplex tapes can achieve around a 5 MBs per second end-to-end transfer rate. Use Volume Mount Analyzer reports to determine how much data will be written to the Virtual Volumes per hour. Convert this value to megabytes per second. Divide the result by the transfer rates achieved to determine the number of physical tape drives required to support Externalization.

Export Tapes

Export tapes are used when you have to send tape data to a government entity or a business partner. They are uncataloged, stacked versions of the original application data set(s) written to CA Vtape. These tapes can be read without CA Vtape being active.

Note: Export Tapes are not used for Disaster Recovery except for a limited number of tapes.

These tapes are created during the Externalization Process. The Backstore Engine first writes to the Primary and Duplex Tapes simultaneously and then writes to the Export Tape. The additional processing step doubles the time it takes to Externalize the data.

Export tapes are generated using the user data set name and format. This requires that the SVTSAS address space have UPDATE or OUTPUT authority to create the data sets. If SVTSAS does not have enough authority, security error messages will be issued and the Export will fail:

```
 SDSF SYSLOG  2827.105 SIV9 SIV9 01/28/1999 LINE  14,177     COLUMNS   1 132
 COMMAND INPUT ===>                                 SCROLL ===> CSR
NC0000000 SIV9     99028 04:24:58.62 ISPDXH1  00000290  SVTS D G
MR0000000 SIV9     99028 04:24:58.67 TSU05815 00000090  SVTSX0600I Cache Size (MB) 00012800/00004920/00004800         000% 419
DR                                   419 00000090             Max Drives=00000010    Threshold=075%
DR                                   419 00000090             Group    Short         Medium        Long          Percent
DR                                   419 00000090             13       00000040(A)   00000000(H)   00000000(H)   000%
DR                                   419 00000090             31       00000040(H)   00000000(H)   00000000(H)   000%
NR0000000 SIV9     99028 04:24:58.67 TSU05815 00000090  SVTSX0100I Command Completed Successfully
N 2800000 SIV9     99028 04:24:59.20 STC05774 00000081  IEC705I TAPE ON 0F2C,208093,SL,NOCOMP,SVTS,SVTS,SVTS.VVE.V899270.PRIMARY
N 2800000 SIV9     99028 04:25:08.98 STC05774 00000090 *TMS001  IEC501A M 0F2F,PRIVAT,SL,NOCOMP,SVTS,SVTS,ISPDXH1.G13.RDBACK
N 2800000 SIV9     99028 04:25:08.98 STC05774 00080094 *IEC501A M 0F2F,PRIVAT,SL,NOCOMP,SVTS,SVTS,ISPDXH1.G13.RDBACK
M 0020000 SIV9     99028 04:25:20.82 STC05774 00000081  ICH408I USER(SVTS    ) GROUP(SYSSTC  ) NAME(TEST                ) 450
D                                    450 00000081    ISPDXH1.G13.RDBACK CL(DATASET ) VOL(205141)
D                                    450 00000081    INSUFFICIENT ACCESS AUTHORITY
D                                    450 00000081    FROM ISPDXH1.** (G)
E                                    450 00000081    ACCESS INTENT(UPDATE )  ACCESS ALLOWED(READ   )
N 0020000 SIV9     99028 04:25:21.53 STC05774 00000090  CAL052SF 08 000 SVTS    /SVTSINI /ISPDXH1. CAS9034E - FUNCTION(RESCHECK)
S                                                        ACCESS TO RESOURCE HAS BEEN DENIED
N 4000000 SIV9     99028 04:25:21.53 STC05774 00000090  IEFTMS50 9XX- 04 SVTS    ,SVTS    ,    SVTS,0F2F,205141,0001,SPDXH1.G13.
S                                                        RDBACK
N 4020000 SIV9     99028 04:25:21.53 STC05774 00000090  IEFTMS50 ***** CA-1 ABEND,F1,04 *****                    **
N FFFFFFF SIV9     99028 04:25:21.61 STC05774 00000090  SVTSP0200E Recover2 routine entered, attempting retry.  See LOGREC for
S                                                        diagnostic information
NC0000000 SIV9     99028 04:25:37.19 ISPDXH1  00000290  SVTS SET BACKSTORE=HOLD,GROUP=13,SUBGROUP=S
NR0000000 SIV9     99028 04:25:37.19 TSU05815 00000090  SVTSX0100I Command Completed Successfully
N 0020000 SIV9     99028 04:25:41.11 STC03107 00000090  VAN0734I Automation Script LOG      EMCCDR  : Control given to the Event
```

Releasing Tapes from Backstore Processing

When an active Externalization task empties a subgroup queue, the Backstore Engine keeps the physical tape and saves the VOLSER, and the last file sequence number used, and the last Virtual Volser written in the Local VCAT. When additional Virtual Volumes for the same group and subgroup are queued for Externalization, the Backstore Engine will request a tape mount for the saved VOLSER and stack the additional Virtual Volumes on that VOLSER. These actions will be repeated until the tape is full, at which time a scratch mount is requested.

There are times when a tape should not be mounted again for additional Backstore processing. For example, when a tape is known to be damaged or when the tape must be taken off-site.

A console command has been provided to release a tape from additional backstore processing. You can use the SVT*n* SET BACKstore=EXCLVOL,VOLser= *volser* console command, where *volser* is the Backstore Tape VOLSER, to release individual tapes. You can use the SVT*n* SET BACKstore, EXCLVOL,Group=*,Type=D console command c to release all DUPLEX tapes for all groups for off-site movement.

When Externalization starts again, the saved PRIMARY VOLSERs are remounted for filling and scratch mounts are requested for the DUPLEXes.

Note: For more information about the SVT*n* SET BACKstore=EXCLVOL command, see the chapter “Console Commands” in the *Administration Guide*.

Limiting the Number of Tape Drives used by the Backstore Engine

The MAXDRIVES setting determines the number of tape drives that the Backstore Engine is allowed to use concurrently. The total number of tape drives in use is the actual count of devices allocated to Recall and Externalization tasks. These values can be displayed by issuing the SVT*n* Display Backstore console command for those CA Vtape subsystems that are executing Recall and Externalization tasks.

When MAXDRIVES is not set to zero, the Backstore Engine only applies the MAXDRIVES limit to Externalization tasks, not Recall tasks.

If starting a new Externalization task would cause the total number of drives in use to exceed the MAXDRIVES setting, the task is rescheduled. Recall tasks are always started, even if the number of drives in use will exceed the MAXDRIVES. This is done to minimize application wait time for a recall.

How the Backstore Engine reacts when the MAXDRIVES setting is exceeded is controlled by the setting for the FullMaxdrivesEnforcement attribute in the <DynamicOptions> Section of parmlib.

FullMaxdrivesEnforcement Set to No

When FullMaxdrivesEnforcement is set to N for no, the drives used by Recall tasks are not counted toward the MAXDRIVES setting. If MAXDRIVES is set to five, five drives can be used by Externalization tasks and any number of additional drives could be used for recall tasks.

Note: If MAXDRIVES is set to zero, the Backstore Engine is not allowed to use tape drives for any reason. No Recall or Externalization tasks will be started..

FullMaxdrivesEnforcement Set to Yes

When FullMaxdrivesEnforcement is set to the default of Y for yes, the drives used by Recall tasks are counted as part of the MAXDRIVES setting. Recall tasks are however still not limited by the MAXDRIVES setting. If MAXDRIVES is set to five, Externalization tasks can use up to five drives. If a Recall task is started and the MAXDRIVES setting is exceeded, an Externalization task is stopped to free its drives and reduce drive usage to the MAXDRIVES setting. If eight Recall tasks are started, all Externalization tasks will be stopped to free their drives and there will be a total of eight drives in use by Recall Tasks. As the Recall tasks complete and release their drives and drive usage drops below the MAXDRIVES setting, the Externalization tasks will be restarted one by one to complete their work.

Note: If MAXDRIVES is set to zero, the Backstore Engine is not allowed to use tape drives. No Recall or Externalization tasks will be started.

Cache Shortage

During a cache shortage, Recall tasks no longer have priority. Externalization tasks are given priority to free up space in cache.

When the Cache In-Use percentage drops to the setting of the CacheWarningThreshold, Externalization tasks no longer have priority.

MAXDRIVES Setting

MAXDRIVES is originally set during the customization process and is local or subsystem specific in scope. Using the SVT*n* SET MAXDRIVES console command, you can adjust the setting dynamically for each CA Vtape subsystem. MAXDRIVES needs to be set for all the subsystems as either zero for Failover Backstore Engine Servers or the available number of tape drives for the Primary Backstore Engine Server.

Since this setting is controlled by a console command, CA Vtape Backstore drives usage can be adjusted as needed. If the demand for tape drives by other applications is high, Externalization can be limited by decreasing the MAXDRIVES setting to make more drives available for the other applications. Conversely, as demand by other applications decreases, Externalization can be allowed to use more drives by increasing the MAXDRIVES setting.

When FullMaxdrivesEnforcement is set to Y, the MAXDRIVES setting is checked by an Externalization task after each Virtual Volume has been copied to a Backstore Tape. When MAXDRIVES is lowered during Externalization, the first Externalization task to complete a copy will detect the change and stop, freeing its tape drives. This process continues until enough Externalization tasks have been stopped to lower the number of tape drives in use to a value equal to or less than the MAXDRIVES setting.

If MAXDRIVES is increased while Externalization is running, the next internal check for work will detect the change and start additional Externalization tasks. The change can be detected earlier if the SVT*n* CHECK BACKstore console command is issued to perform an immediate check for work after increasing the MAXDRIVES setting.

You can display the current setting of MAXDRIVES by using the SVT*n* Display Backstore and Display Groups console commands. The ISPF Interface Group Lists option also displays the MAXDRIVES setting.

Automatic Recall Source Switching

The IO engine of CA Vtape reads and writes Virtual Volumes from the DASD Buffer (cache). The Backstore Engine stacks the Virtual Volumes to a Primary and optional Duplex dataset on tape. When USS Backstore is active, the Backstore Engine writes and reads Virtual Volumes to the Triplex copy on USS. These Backstore copies are the Recall Sources. When the licensed CA Vtape Peer-To-Peer Option is configured, Backstore copies on Remote systems are included as additional Recall Sources.

Automatic Recall Source Switching improves data availability by automatically switching between Recall Sources when a recall failure occurs. This feature is only available when Dynamic Cache Management is active.

When this feature is active CA Vtape maintains counters for the Recall Sources and establishes a hierarchy for automatically moving through the Sources when a recall attempt fails. The order in which the Sources are used can be influenced by the SVT*n* SET RECALL console command and the OffsiteBackstoreCopy and RecallAttemptsThreshold attributes in the Group Definitions.

Note: If the last Source becomes excluded, then the Virtual Volume is mounted empty, and the application will take an SX13 ABEND.

The OffsiteBackstoreCopy attribute determines if any Backstore copies should be excluded from Recall processing. This attribute facilitates sending copies of Virtual Volumes offsite so only the available copy is considered for Recalls.

RecallAttemptsThreshold limits the number of times a Recall is attempted from a single Source. Once the threshold is exceeded, if another eligible Source is present, Recall will attempt to use the alternate Source. Once all Sources have exceeded the threshold, the volume will be mounted as an empty Virtual Volume. If the attribute is set to zero, automatic switching is disabled. The Recall will always use the Backstore copy after applying the OffsiteBackstoreCopy exclusion rule.

We recommend setting RecallAttemptsThreshold=2.

Each time a recall is required the Integrated Catalog Facility (ICF) is interrogated for Primary and Duplex dataset entries. When USS Backstore is active, the USS file system is also interrogated regardless of whether the group is set to write Triplex copies. This is because the group may have been changed over time. The OffsiteBackstoreCopy works by excluding from consideration the data set entries that have been sent off-site. Next, the RecallAttemptsThreshold counters for the eligible Sources are examined. Any Source whose threshold has been exceeded will be eliminated from consideration. The recall order set by the SVT*n* SET RECALL console command will then be considered.

When the licensed CA Vtape P2P Option is configured, local Sources take precedence over remote Sources to minimize network traffic.

Allocation Logic - CA MIM or IBM GRS Support

There are different configurations for physical tape devices supported by CA Vtape, as follows:

- Sysplex environments using CA MIM or IBM GRS.

  In these environments, the physical tape devices are truly shared and all systems see them as ONLINE at the same time. If one system is using a particular physical tape, CA MIM or IBM GRS will exclude it from allocations in all the other systems. This environment is automatically supported by CA Vtape if you specify the VTPARMS parmlib attribute BypassOfflinePhysicalDevices=Y (default value).

- Nonsysplex environments using CA MIM or IBM GRS to implement tape sharing.

  In these environments a pool of physical tape devices is shared between systems by keeping the physical devices in ONLINE status only in the system using it and in OFFLINE status in all the others. When allocations occur and the operating system finds all the required drives are OFFLINE, WTOR messages are sent to the console. In these environments, these messages are automatically replied to with WAIT/NOHOLD, which signals the sharing software to look for an unused physical tape device and vary it ONLINE in the system requesting the allocation. This environment is automatically supported by CA Vtape if you specify the VTPARMS parmlib attribute BypassOfflinePhysicalDevices=N.

- Nonsysplex environments where the physical tape devices are not shared.

  In these environments, each LPAR has dedicated physical tape drives in ONLINE status. This environment is automatically supported by CA Vtape if you specify the VTPARMS parmlib attribute BypassOfflinePhysicalDevices=Y (default value).

BypassOfflinePhysicalDevices=Y

With BypassOfflinePhysicalDevices set to Y, CA Vtape starts an Externalization or Recall task when all the required tape devices for that task are found available and in ONLINE status. If there are not enough tape drives available, the task is terminated, rescheduled, and automatically retried without any operator intervention. The following are typical messages issued under these conditions:

- SVT*n*PX116E No drives available.
- SVT*n*PX020W Physical drive allocation failed. Task will be terminated and rescheduled.

If the required tape drives will not be available for long periods of time, the operator should issue the SVT*n* SET MAXDRIVES console command to lower the maximum number of tape devices CA Vtape is trying to use.

If the task being retried is more important than Externalization processes already in progress (for example, a Recall), you can stop one or more of the existing tasks gracefully by issuing the SVT*n* STOP PTASKID or SVT*n* STOP GROUP console commands.

Note: For more information about these commands, see the chapter “Console Commands".

If the FullMaxdrivesEnforcement attribute in the DynamicOptions Section of the parmlib is set to Y, an Externalization task will automatically be stopped when a Recall causes the MAXDRIVES setting to be exceeded.

BypassOfflinePhysicalDevices=N

With BypassOfflinePhysicalDevices set to N, CA Vtape will consider both ONLINE and OFFLINE devices for allocations. Use this option only in those environments using CA MIM, IBM GRS or equivalent software packages to automatically vary devices ONLINE as needed, or with tape drives dedicated to each LPAR. CA Vtape will drive an allocation for a tape device as long as the required tape devices for the esoteric defined in the group attributes (or for the device type stored in the ICF catalog for Recalls) are defined to the operating system. The allocation will be driven regardless of the devices' ONLINE/OFFLINE status.

Note: After WAIT/NOHOLD has been replied to the operating system WTOR messages, no further allocation or deallocation processes will be executed by the Backstore Engine until the required device becomes ONLINE.

Tape devices that the Backstore Engine already has allocated, will not be automatically released at task termination if one or more allocations are pending. The operator can manually stop a task that is waiting for a tape device to release the wait if for some reason CA MIM/IBM GRS is not responding properly. For customers implementing this option, it is very important to properly set MAXDRIVES to a value that will guarantee availability of tape devices when needed.

Contention Over Backstore Tapes

The following contention scenarios involve the same Backstore Tape:

- A Recall is attempted and the Backstore Tape is in use by Recycle on any system. **Expected action:** The Backstore Tape will be released as soon as the current Virtual Volume being processed is completed.
- A Recall is attempted and the Backstore Tape is in use by Externalization on the same system. **Expected action:** The Recall will be queued under the Externalization task and the queue will be processed in FIFO order. The queue can be displayed by using the SVT*n* DISPLAY BACKSTORE console command.
- A Recall is attempted and the Backstore Tape is in use by Externalization on a different system. **Expected action:** The Recall task will enqueue the cartridge using SYSZVOLS and will wait until the resource is released.
- A Recall is attempted and the Backstore Tape is in use by another Recall recalling the same Virtual Volume on any system. **Expected action:** The Recall task will issue message SVT*n*V0400I and wait for the other Recall to complete. The job that requested the Recall will access the Virtual Volume from the DASD buffer as soon the enqueue on SYSZVOLS for the Backstore Tape is released.
- A Recall is attempted and the Backstore Tape is in use on the same system for another Recall. **Expected action:** The Recall will be queued under the in-progress Recall task and the queue will be processed in FIFO order. You can display the queue by using the SVT*n* DISPLAY BACKSTORE console command. If several Recalls are started at the same time, one or more of them may have originally started as separate tasks, but as soon the situation is detected CA Vtape will automatically requeue all the Recalls under one task.
- A Recall is attempted and the Backstore Tape is in use on a different system for another Recall. **Expected action:** The Recall task will enqueue the cartridge using SYSZVOLS and will wait until the resource is released.
- An Externalization is attempted and Recycle is in progress for the same group or subgroup in any system. **Expected action:** Externalization will not start until Recycle finishes or is stopped.

- An Externalization is attempted and Recall is using the Backstore Tape on the same system. **Expected action:** Message SVT*n*PX132W will be issued and the Externalization task will be rescheduled.
- An Externalization is attempted and Recall is using the Backstore Tape on a different system. **Expected action:** The Externalization task may enqueue the Backstore Tape using SYSZVOLS and wait until the resource is released. If not, message SVT*n*PX132W will be issued and the Externalization task will be rescheduled.

By design, each Externalization subtask on each system uses its own Backstore Tapes for Externalization. There is no potential contention for the same Backstore Tape between Externalization tasks on the same or different systems.

Note: Setting up one subsystem as both the Externalization Server and as the Recall Server eliminates all cross-system contention for Backstore Tapes. By propagating the SVTSX and SVTRCYCL QNAMES across systems, contention for Backstore Tapes with Recycle can be eliminated.

USS Backstore

The CA Vtape USS Backstore feature utilizes z/OS UNIX System Services (USS) for reading and writing Virtual Volumes during Recall and Externalization. USS Backstore supports any USS file systems but is primarily intended for use with Network File Systems (NFS).

USS Backstore seeks to augment the Backstore engine to allow a copy of a Virtual Volume to be written to a USS file system server. A NFS server, while not required, can be a data de-duplication and or replication device.

USS Backstore will enable:

- Access to less expensive disk
- Tapeless implementations
- Access to data de-duplication and replication devices to conserve network bandwidth

USS Backstore requires an active TCP/IP stack, USS, and the z/OS NFS Client running on your zOS/390 system. If necessary, refer to IBM's zOS/390 documentation regarding how to install or setup TCP/IP, USS, and the NFS Client on your system.

How the USS Backstore Works

CA Vtape continues to use existing high speed mainframe DASD for Virtual Drive emulation in the IO Engine.

CA Vtape has added an optional USS Backstore copy attribute for Virtual Volumes that can be used in externalization policies defined by Group definitions through parmlib. This is in addition to the Primary and Duplex attributes. The USS Backstore copy is defined by the Triplex attribute.

Note: Triplex is mutually exclusive with the Export attribute meaning that if you define an Export attribute value you cannot define a Triplex value or vice versa.

Primary and Duplex copies Virtual Volumes to and from physical tape while Triplex copies them to and from USS files. The possible combinations of Backstore Copies are:

- Primary only
- Primary and Duplex
- Triplex only
- Primary and Triplex
- Primary, Duplex and Triplex

You can use the USS file system Triplex copy in addition to their traditional Primary and Duplex copies on physical tape. Or you can use the USS file system Triplex copy as a way to eliminate physical tapes. The USS file system can write to NFS servers which can be located in a cold DR site and can be a data de-duplication device that supports NFS. Any NFS server supported by z/OS can be used. CA Vtape does not provide replication or data de-duplication directly but can exploit these features if provided by the NFS server. Most data de-duplication NFS servers also offer data replication.

With the CA Vtape flexible policy based grouping, performance, replication, compression, and physical tape requirements can be assigned to suit the application.

During externalization, the group attributes determine how many tape drives are allocated and whether a Triplex file is to be created.

During recall, if USS Backstore is enabled, then any located Triplex copy is considered as an additional recall source. This is true whether the group currently has Triplex specified or not. The group attributes OffsiteBackstoreCopy and RecallAttemptsThreshold have been extended to consider the Triplex copy along with SET RECALL order operator command.

Note: For more information, see the section Automatic Recall Source Switching (see page 216) and the section Recall Sources in the *Administration Guide*.

Implementation Required Skills

Successful implementation of USS Backstore requires various skills and is a team effort.

Do not attempt to implement USS Backstore until you have arranged access to people with skills in the following areas:

- z/OS UNIX System Services (USS) and file systems, to set up a functional UNIX and file systems environment.
- Security server (for example, CA ACF2 for z/OS, CA Top Secret for z/OS, or IBM RACF) or the security product you use, to authorize access to resources.

USS File Systems

This section provides additional information about setting up your z/OS UNIX System Services (USS) and file systems.

USS Path Setup

CA Vtape can use HFS, zFS or NFS file systems for its general usage. Before you run USS Backstore, your USS administrator must set up a directory path and optionally configure a file system.

Note: We recommend using zFS file system for the initial path.

During operation, CA Vtape dynamically mounts additional file systems. File systems are mounted during startup, and as required in response to SVTS operator refresh commands.

The following example shows the structure of the paths:

```
/u/users/vtape
     signature
          triplex
               group01
               group02
               group03
               group04
               … (groups 11-14, 21-24, 31-34, 41-44, 51-54 and 61-64)
               group71
               group72
               group73
               group74
```

/u/users/vtape

Is the primary mount point or directory and is required to exist before starting CA Vtape with USS Backstore enabled. This directory name may be customized by changing the RootDirectory attribute in the <StartupOptions> section of parmlib. This name does not need to match between SVTS subsystems in the same CA Vtape Complex.

/u/users/vtape/signature

Signature is an unique hexadecimal value assigned to the CA Vtape Complex. This signature directory name is used to guard against multiple CA Vtape Complexes accidentally accessing incorrect files.

Note: The signature value assigned to the CA Vtape Complex can be displayed by the SVT*n* Display Status console command.

/u/users/vtape/signature/triplex

Is a required directory and mount point that is dynamically created and mounted by CA Vtape during startup and parmlib refresh if the file system must be changed. This directory and mount point typically refers to a NFS file system under which the USS Backstore copies will be stored in sub-directories.

/u/users/vtape/signature/triplex/group*xx* (where *xx* is one of the 32 groups)

Are dynamically created directories used to hold the USS Backstore copies. To accommodate spreading data across multiple NFS servers, each group directory can optionally be customized as an individual mount point.

Security Considerations

The user ID assigned to CA Vtape started tasks, normally SVTS and SVTSAS procedure, as well as the batch utilities that require access to USS and must all have a valid OMVS segment. Your security administrator must set up these segments.

To set up an OMVS segment

1. Choose an OMVS UID number to associate with your user ID. Your security administrator may have a policy for assigning OMVS UID numbers. If not, use a unique number.

   Note: For more information about OMVS UID numbers, see IBM's UNIX System Services Planning guide.

2. Define the OMVS segment for the user. For a user ID *uuuuuuu*, UID number *nnn*, and home directory path_name, enter the following commands:

   - For CA ACF2 for z/OS systems, enter the following commands:

     ```
     SET PROFILE(USER) DIV(OMVS)
     INSERT uuuuuuu UID(nnn) HOME(path_name) OMVSPGM(/bin/sh)
     ```

- For CA Top Secret for z/OS systems, enter the following commands:

  ```
  TSS ADD(uuuuuuu) HOME(path_name) OMVSPGM(/bin/sh) UID(nnn)
  GROUP(ggggggg)
  ```

- For RACF systems, enter the following command:

  ```
  ALU uuuuuuu OMVS(UID(nnn) HOME(path_name) PROGRAM(/bin/sh))
  ```

Note: The OMVS segment must contain the following:

- A home directory (HOME)
- A login shell (PROGRAM or OMVSPGM)

3. Complete this process for each user ID that you want to authorize. To confirm the contents of the OMVS segment enter the following commands:

   - For CA ACF2 for z/OS systems, enter the following commands:

     ```
     SET PROFILE(USER) DIV(OMVS)
     LIST uuuuuu
     ```

   - For CA Top Secret for z/OS systems, enter the following command:

     ```
     TSS LIST(uuuuuu) DATA(ALL)
     ```

   - For RACF systems, enter the following command:

     ```
     LISTUSER uuuuuu OMVS NORACF
     ```

4. Choose a home directory to associate with each user ID, and ensure that it exists and that the UID has read/write access to it. You can use the UNIX directory (*path_name*) as shown in Step 2, or you can use a customized home directory name.

   For example, to set up a directory called /u/users/vtape for UID*nnn*, issue the following commands in the OMVS UNIX shell:

   ```
   mkdir /u/users/vtape
   chown nnn /u/users/vtape
   chmod 775 /u/users/vtape
   ```

5. Confirm the owner and access to the directory by using the following command:

   ```
   ls -ld /u/users/vtape
   ```

   The following result appears:

   ```
   drwxrwxr-x   2 user  group  8192 Sep  31 14:58 /u/users/vtape
   ```

Configure the USS Backstore File System

Assuming that you have an active TCP/IP stack running on your z/OS system, activating USS Backstore consists primarily of starting CA Vtape after configuring some additional parmlib members and attributes.

Overview

The following are the basic parmlib configuration steps.

1. Setup the VUSSMNTS (VTape USS Mounts) member. This member defines the USS mount points. The sample member VUSSMNTS provided in the PARMLIB library includes a description of each attribute within this member. The <Triplex> section is required and all others are optional.
2. Setup the groups definitions in the VTGROUPS member. Groups that will write a USS Backstore copy must add the Triplex=USS attribute.

   Note: The Triplex attribute is mutually exclusive with the Export attribute.
3. Define the names of the VUSSMNTS member in the parmlib <Directory> section of VTPARMS member. Uncomment and add the USSMountPoints attribute to the parmlib <Directory> section. This attribute names the parmlib member containing the VUSSMNTS attributes.
4. Within the <StartupOptions> section add UnixSystemServices = Y and customize the RootDirectory = '/u/users/vtape' attribute.

   Note: The RootDirectory must already exist.

Detailed Configuration Instructions

In order to insure proper Backstore Primary and Failover Server functionality, each SVTS subsystem in a CA Vtape Complex must share the same USS mount points. To facilitate this, we recommend sharing the same VUSSMNTS parmlib member.

Note: It is not required for the <StartupOptions> RootDirectory = '/u/users/vtape' attribute to match.

To configure the USS Backstore file system

1. Customize the VUSSMNTS member to include the <Triplex> section is at a minimum.

```
<USSMountPoints>

<Triplex>
 DSName  = VTAPE.TRIPLEX
 FSType  = NFS
 PARMS  =  10.130.42.54:'            + ip address
           '"/nas/vtape/triplex"'    ; drive directory
```

   Observe the following:

   - The DSName attribute depends on the FSType attribute. When FSType is either HFS or ZFS then DSName is the actual dataset name. When FSType is NFS, then DSName is a name selected by you to represent the file system.
   - The FSType attribute indicates the type of file system to be mounted. The valid settings are HFS, ZFS or NFS.

- The PARMS attribute is intended for NFS file systems and conveys the NFS server name or TCPIP address along with the path name to be mounted on the NFS server. Note that other NFS mount attributes may be added to the PARMS attribute. For more information, see the IBM documentation *z/OS Network File System Guide and Reference*.

2. Setup the VTGROUPS parmlib member and indicate what combinations of Backstore copies are desired at the group level:

```
<GroupDefinitions>

  Group01 = USS_Only
  Group02 = Primary_Duplex
     ...
  Group74=

<USS_Only>
  Description='USS Only'
  MB_Threshold               = 2000
  ShortRetention             =    7
  MediumRetention            =   21
  Triplex                    =  USS
  RecallAttemptsThreshold    =    2
```

   Observe the following:

   - The attribute Triplex=USS can be assigned to as many of the 32 groups as desired.
   - Other valid combinations are:
     - Triplex only
     - Primary only
     - Primary and Duplex
     - Primary and Triplex
     - Primary, Duplex and Triplex

3. Change the parmlib directory section to include the new VUSSMNTS member. We recommend sharing the VUSSMNTS member with all SVTS subsystems in the same CA Vtape Complex.

```
<ParmlibDirectory>
  USSMountPoints     = VUSSMNTS
```

4. Change the < StartupOptions> section to include the new UnixSystemServices and RootDirectory attributes.

```
<StartupOptions>
     UnixSystemServices=Y
     RootDirectory='/u/users/vtape'
```

5. Start the SVTS subsystem.

   When starting with the USS mount point section for the first time, the directory structure is built and the file systems are mounted dynamically. The file system attributes are saved in the control files to be used on subsequent restarts to insure the file system settings do not change without confirmation. Additionally, a timed routine checks to insure the file systems are mounted properly every 5 minutes.

Displaying the File System Mount Points

To display file system information use the SVTS DISPLAY USS console command. To return only mount point information use SVTS DISPLAY USS,M. The following example output is returned:

```
SVT7X2105I Unix File System Information
Data Set   OMVSUSR.VTAPE
Type       ZFS,R/W,Active
MountPoint /u/users/vtape
Statistics           Capacity(gb) Use%
InUse                         .00   0%
Free                          .16 100%
Total                         .16
Data Set   VTAPE.TRIPLEX
Type       NFS,R/W,Active
MountPoint /u/users/vtape/49C17EC0/triplex
Parms      10.130.42.54:"/nas/vtape/triplex",hard,llock(y),proto(tcp)
           ,xlat(N),attrcaching(y),datacaching(y),delaywrite(32),readah
           ead(8)
Statistics           Capacity(gb) Use%
InUse                         .44   1%
Free                        32.83  99%
Total                       33.27
SVT7X0100I Command Complete
```

Changing File System Mount Points

It is possible to change the file system mount points without restarting the SVTS subsystems but we recommend taking special precautions when doing so.

Note: For a detailed description how to accomplish this task, see the section File System Reconfiguration (see page 230).

USS Backstore Advanced Configurations

The various ways USS Backstore can be configured depends on how you customize your mount points in the VUSSMNTS parmlib member and whether the NFS server is capable of replication.

Customizing Mount Points

At a minimum, the Triplex directory is required to be a mount point. Without further customization, USS files written to the 32 different group directories are part of the Triplex file system.

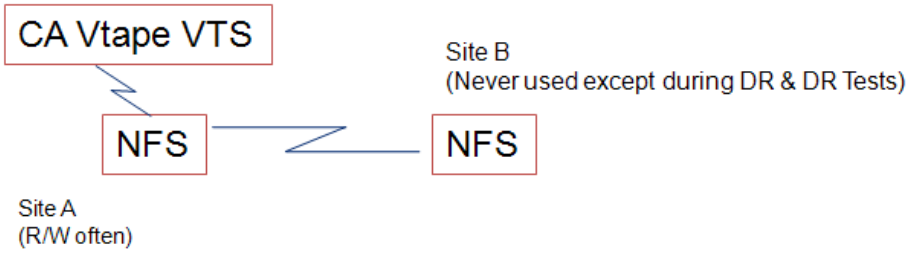
However, to facilitate distributing file space and server capacity, each of the 32 different group directories can also be mount points to different NFS servers.

We believe a common configuration will involve defaulting most group directories to the Triplex file system, and customizing a few groups to other NFS servers.

Exploiting NFS Server Replication

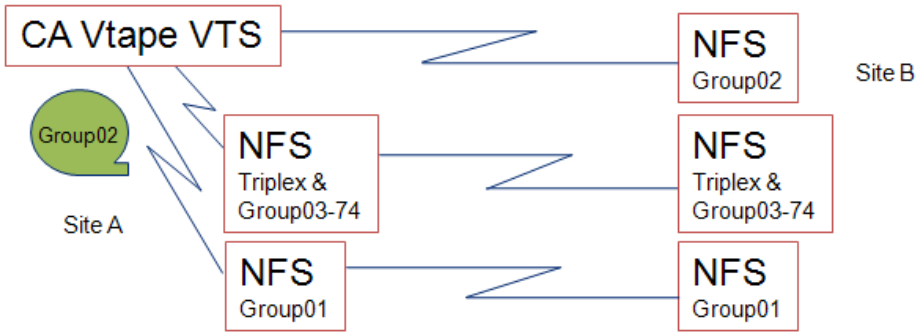
CA Vtape does not directly control replication but can utilize NFS servers that do implement replication. Control over replication is managed by the NFS server.

The following is an illustration example of a simple Triplex only mount point:



During normal operations the NFS Backstore copies are created on Site A and replicated to Site B. Site A's Backstore copies can also be used for recalls. During DR testing or an actual DR event, the replication mirror between Site A and Site B is turned off. An LPAR on Site B will run an instance of CA Vtape. All DR activity can take place reading and writing Virtual Volumes. When the test is complete the replication can be re-established between Site A and Site B. Anything written on Site B will be lost and changes made to Site A will be copied over to Site B. In this scenario CA Vtape does not participate in the reconfiguration, it is handled by the NFS devices.

Now consider the following possible configuration:



This example illustrates an advanced configuration exploiting additional group mount points along with some NFS servers replicating data and some NFS servers without replication.

In the example, the Triplex file system contains groups03-74 and are replicated between Site A and Site B. Group01 has been given its own dedicated NFS server which is also replicated between Site A and Site B. Finally, group02 has its own NFS server but resides in Site B so the group definition has been setup to write a Primary tape at Site A. Note that writing to group02 requires spanning the entire wide area network and throughput is slower than when writing to the NFS servers locally.

NFS Performance

How you setup connectivity to your NFS server plays a key role in determining throughput. You should know your environment and determine what equipment is between the z/OS host and the NFS server.

z/OS Connectivity

Most common network adapters will be OSA Express or LCS adapters. A typical OSA adapter contains between two and four 1 GbE or 1000 BASE-T connections. Some OSA adapters have a single 10 GbE connection. The 1 GbE and 1000 BASE-T connections are capable of up to 125 MB/sec and the 10 GbE is capable up to 1250 MB/sec.

In comparison, a single 3592 is able to write between 60-100 MB/sec. However, most customers are writing with drives at 20-50 MB/sec.

CA Vtape is able to consolidate Backstore activity to just a few Backstore Primary and Failover servers. This means the z/OS connectivity required by the z/OS NFS Client can be consolidated to an LPAR or two. The connectivity required by the z/OS NFS Client can be shared among other z/OS network applications. Adequate network adapter capacity may already exist in your environment.

Location of the NFS Server

Where you decide to locate the NFS server is dependent on many factors but most important is whether replication is available on the NFS server.

With replication enabled on the NFS server we recommend locating the NFS server as close to the z/OS host as possible. Locating both on the same LAN segment is best because large packet sizes can best be exploited.

Typically, the replica is sent by the NFS server on a separate network adapter which should be on a different network segment best suited for communicating with the replica NFS server.

If the NFS server is enabled for data de-duplication then only unique data is sent to the replica NFS server resulting in conserving disk space on both servers and network capacity.

Without replication enabled on the NFS server means only one copy of a Virtual Volume will be stored on the NFS server.

Note: We recommend you insure an additional Backstore copy be created on Primary physical tape. The placement of the NFS server can be in the same site or can be located in a different site. The expected throughput will diminish the further the NFS server is located away from the z/OS host. A general rule of thumb is add a millisecond of response for every 100 miles between sites.

File System Reconfiguration

Once the USS Backstore file system has been setup there should be little need to make changes to the configuration. However, should the need arise we recommend careful planning and to implementing file system changes during periods of low activity.

The underlying approach is to insure the IO Engine disk cache has enough capacity to tolerate suspending Externalization to the USS file systems and that applications requiring recalls tolerate mount pending for the time during the outage.

We recommend you implement Backstore Primary and Failover servers as described in section Configuring the Backstore Engine in Primary and Failover Roles (see page 211).

To implement file systems changes

1. Prevent scratch synchronization jobs from running during the reconfiguration.
2. Shutdown any non-essential CA Vtape subsystems.
3. For each active CA Vtape subsystem in the CA Vtape Complex do the following:
   - Issue SVTS D G and note the current Max drives and USS settings
   - Issue SVTS SET MAXUSS=0
   - Issue SVTS SET MAXDRIVES=0

4. Make changes to the VUSSMNTS parmlib member to reflect the desired configuration changes.
5. For each active CA Vtape subsystem in the CA Vtape Complex do the following:
   - Issue SVTS REFRESH=USS
   - Reply Y to activate the parmlib changes

   At this point the new file system configuration is active but some of the Virtual Volume files are unavailable.
6. Copy the files from the original directories to the new directories.

   This can be done by mounting the original directories to temporary mount points and then copy to the new directory.
7. For each active CA Vtape subsystem in the CA Vtape Complex reset the Max drives and USS to the original settings noted in step 3.
   - Issue SVTS SET MAXUSS=*n*
   - Issue SVTS SET MAXDRIVES=*n*

Chapter 15: CA Vtape P2P Option

This section contains the following topics:

Overview and Requirements (see page 233)
How P2P Works (see page 234)
Scratch Mount Requested by a Remote Virtual Tape Unit (see page 234)
P2P Security Considerations (see page 235)
Configure for CA Vtape P2P Subsystems (see page 236)
Set Up a Simple P2P Configuration to Test TCP/IP Connectivity (see page 237)
Set up a SVTS Subsystem in a Separate CA Vtape Complex (see page 241)
Test Remote Virtual Volumes Access (see page 243)
CA Vtape P2P Advanced Configurations (see page 250)
Scratch Processing for Remote Virtual Volumes (see page 251)

Overview and Requirements

The CA Vtape P2P Option is an optional component which requires a separately licensed LMP key. CA Vtape P2P uses TCP/IP to provide real time remote duplexing of Virtual Volumes.

Virtual Volumes can be shared by multiple SVTS subsystems in a SVTS complex environment without CA Vtape P2P. A SVTS complex is comprised of several SVTS subsystems accessing a common set of control files and Virtual Volumes through the use of shared DASD across one or more LPARs. This configuration also works well when all of the subsystems are running in a common physical location.

CA Vtape P2P allows separate SVTS complexes with single or multiple SVTS subsystems to communicate and share a common subset of Virtual Volumes using TCP/IP. These systems do not need to share DASD or a common set of control files. Additionally these SVTS subsystems may be in physically diverse locations.

CA Vtape P2P requires an active TCP/IP stack running on your zOS/390 system. If necessary, refer to IBM's zOS/390 documentation regarding how to install or setup TCP/IP on your system.

CA Vtape P2P is only supported for Virtual Volumes created by a CA Vtape Subsystem running with CacheManagement=Dynamic.

How P2P Works

CA Vtape P2P is a true peer-to-peer implementation utilizing TCP/IP to provide communication across SVTS subsystems. Today, separate SVTS subsystems running autonomously can each have their own set of Virtual Volser ranges, Virtual Devices, and control files. CA Vtape P2P allows these SVTS subsystems to further define remote subsystems, remote Virtual Drives, remote Virtual Volumes, remote groups, and remote data filters. This gives the SVTS subsystems the added ability to share a common overlapping subset of Virtual Volumes without the need of shared DASD or control data sets.

CA Vtape P2P introduces several new concepts which would be useful to review here. For purposes of this discussion, consider what happens when an application requests a scratch tape be mounted on a remote tape drive. Further, consider calling the system where that application is running the *local system* and the remotely connected system the *remote system*. Observe the following:

- CA Vtape P2P uses the concept of *local versus remote Virtual Tape Units (VTU) or drives*. From CA Vtape you are already familiar with *local Virtual Tape Units*, these are the same Virtual Devices you should already be accustomed to using without the need of utilizing CA Vtape P2P. These devices do not communicate with *remote systems*. In contrast to a *local VTU*, a *remote VTU* has all of the same features along with the added capability of being able to establish a TCP/IP connection with a remote system.
- Just as *local VTUs* are assigned by data class or data set filters defined within the parmlib, so too are *remote VTUs* assigned in a similar manner utilizing *remote data set* or *data class filters*.
- *Remote data set* or *data class filters* point to a *remote group definition*. The *remote group definition* contains the name of the remote system with whom to communicate. Further, the remote group definition also contains a Virtual Volume pool name which describes Virtual Volser ranges from which to assign a *remote Virtual Volume*.

Scratch Mount Requested by a Remote Virtual Tape Unit

When scratch mount is requested by a remote Virtual Tape unit the following occurs:

- Initially, the *remote VTU* will request a connection with the remote system based on the name defined within the *remote group definition*.
- When a connection request is accepted, the *remote system* will allocate a *Client Drive*. The *Client Drive* will be used to mirror data sent to it by the *local system*.
- The *local system* then creates a list of valid Virtual Volser ranges as defined in the Virtual Volume pool which was assigned by the *remote group definition*. The Virtual Volser range list is then transmitted to the *remote system* where it will be used to select a scratch volser.

- The *remote system* uses its own Virtual Volume POOL8 and attempts to select an appropriate volser based on the list provided by the *local system*.

  If no compatible scratch volser is available, the *remote system* will then attempt to dynamically assign one hundred available scratch volsers to its own POOL8 from within one of the volser ranges as originally requested by the *local system*.

- Once a scratch volser has been selected, the *remote system* will simulate the mount on the *Client Drive*. The scratch volser is then transmitted back to *remote VTU* running on the local system.

- After receiving information about the selected Virtual Volume, the *local system* completes the mount which then signals the application that it can begin writing data.

Important! At this point the same Virtual Volser is mounted on remote Virtual Tape units on both the local and remote system.

- As the application writes data onto the Virtual Volume, the *remote VTU* running on the *local system* records that data to DASD cache and simultaneously transmits that data to the *Client Drive* running on the *remote system*. There, the data is recorded separately onto the DASD cache of the *remote system*.

  This effectively allows both systems to record the same application data to their respective Virtual Volume cache data sets at the same time, thereby duplexing the content of the Virtual Volume.

- When the volume is unloaded, the remote connection is closed and both systems will then contain their own copy of the Virtual Volume in their respective caches.

- Finally because the Externalization processes on these systems are autonomous, each will then process the Virtual Volume cache data set based on their own respective group definition. Group definitions have a comprehensive set of attributes which may be used to control where and how many externalized copies of the Virtual Volume are created.

P2P Security Considerations

The SVTS and SVTSAS started tasks must be assigned an OMVS segment in your security system to allow CA Vtape to use TCP/IP under UNIX System Services. Consult your security product manuals to determine the appropriate method to define and assign an OMVS segment.

Configure for CA Vtape P2P Subsystems

Assuming that you have an active TCP/IP stack running on your zOS/390 system, activating CA Vtape P2P consists primarily of starting CA Vtape after configuring some additional parmlib members and attributes.

To configure parmlib

The following are the basic parmlib configuration steps.

1. Setup the VTP2POPT (*VTape P2P OPTions*) member. This member defines basic TCP/IP and CA Vtape P2P processing options. The sample member VTP2POPT provided in the PARMLIB library includes a description of each attribute within this member. The default values for most attributes should suffice in many cases.

   Note: Keep in mind that this member is where you will describe the parameters used by the *local system* to establish the TCP/IP P2P Listener. The P2P Listener processes connection requests from *remote systems*.

2. Setup the VTP2RMTS (*VTape P2P REMotes*) member. This member defines *remote systems* that are allowed to communicate with this *local system*. The sample member VTP2POPT in the PARMLIB library includes a description of each attribute within this member.

   Important! Only the remote systems defined in this member are allowed to communicate with this local system. This member effectively names each remote system and describes the TCP/IP attributes necessary to request and accept connections with that remote system.

3. Define the names of the VTP2POPT and VTP2PRMT members in the parmlib <Directory> section of VTPARMS member. You will need to uncomment and add both the PeerToPeerOptions and PeerToPeerRemotes attributes to the parmlib <Directory> section. These attributes name the parmlib members containing the VTP2POPT and VTP2PRMT attributes.

4. If you have not done so already, you will also need to know how to setup Virtual Volume Pooling using the VTPOOLS parmlib member. Virtual Volume Pooling is a requirement of CA Vtape P2P. It allows CA Vtape P2P connected systems to effectively share subsets of Virtual Volumes through volser ranges.

5. Setup *remote Virtual Tape Units* (VTUs) or drives in the VTDRIVES member. Remote drives are only assigned when mounting a remote Virtual Volume. Likewise local drives (*local VTUs*) are only assigned for mounting local Virtual Volumes.

6. Setup *remote groups* definitions in the VTGROUPS member. *Remote groups* are just like *local groups* but utilize a few additional attributes, which are:

   - RemoteSystemPrimary
   - RemoteSystemSecondary
   - ExchanceMetadataOnly

7. The attributes which distinguish a group definition as a *remote group* definition are RemoteSystemPrimary, RemoteSystemSecondary, or both. These attributes specify the name of the *remote system* with whom to communicate when the allocation or scratch mount of a data set associated with this group is requested.

   Note: A good description of these additional attributes can be found in the VTGROUP member of the CCUUPARM library.

8. Setup *remote data set* or *remote data class* filters or both in the VTFILTR member. These filters are used for scratch mounts and define how to associate data sets with *remote groups*.

Set Up a Simple P2P Configuration to Test TCP/IP Connectivity

CA Vtape P2P normally requires more than one SVTS complex. Several SVTS subsystems on one or more LPARs, sharing the same set of GLOBAL and BSDS1 data sets, are all considered part of the same SVTS complex. Separate SVTS complexes or subsystems require separate sets of DASD cache, and GLOBAL and BSDS1 control files. Learning to use CA Vtape P2P in this type of environment may seem complex. We therefore recommend starting with a small CA Vtape P2P parmlib configuration designed simply to test TCP/IP connectivity. This configuration will only consist of a single SVTS subsystem and will not require any *remote virtual tape unit* (VTUs), *groups, or data filters*.

As stated previously, the purpose of this configuration is to test easily that you can properly define CA Vtape P2P and establish connectivity through TCP/IP. To test TCP/IP connectivity through a single system you will be utilizing the TCP/IP loopback IP address 127.0.0.1. This will allow you to configure the SVTS subsystem as a *remote system* to itself. While you will not be able to test a remote Virtual Tape mount, you will be able to test connectivity by utilizing remote SVTS commands and thereby confirm that CA Vtape P2P connectivity is working properly with your TCP/IP stack.

To set up a simple CA Vtape P2P configuration

1. Configure the VTP2POPT parmlib member. While most of the default attributes will suffice, the most important attributes to consider in this member are:

   TCPNAME=$NULL

   This attribute is normally assigned the job or task name of the TCP/IP stack running on your zOS/390 system.

   ListenOnPort=1100&SVTS(-1:1)

   This attribute assigns the port number used by the P2P Listener. The default value assigns 1100*n* where *n* is the SVTS subsystem number, that is, from 1 to 8 (effectively 11001 to 11008).

   Note: If these port numbers conflict with other numbers on your system you can change the port number prefix, that is, change the number to something like 1200&SVTS(-1:1)

2. Configure the VTP2PRMT parmlib member. For now we recommend simply defining the local system as a remote system. This will allow you to test connectivity by sending remote commands to the local system.

   In VTP2PRMT define the local system as remote system named localhost, for example:

   ```
   <PeerToPeerRemotes>
       RemoteSystem=LocalHost

   <LocalHost>
      Port = 1100&SVTS(-1:1)
      IPAddress = 127.0.0.1
      ConsoleSuffix = LCL&SVTS(-1:1)
   ```

   In this case the port number must match the ListenOnPort attribute value defined in the VTP2POPT member.

   Note: If you specified a ListenOnIPAddress attribute in VTP2POPT, then the IPAddress must also match that attribute. If you did not specify a ListenOnIPAddress attribute then the loopback address, 127.0.0.1 should work.

   In this example, we have chosen a ConsoleSuffix of LCL*n* where *n* is the SVTS subsystem number 1 to 8. The ConsoleSuffix is a user defined four character value used to uniquely identify the remote system when issuing remote SVTS commands or displaying remote console text.

3. In the parmlib <directory> section of VTPARMS, configure the PeerToPeerOptions and PeerToPeerRemotes attributes to point at VTP2POPT and VTP2PRMT members.

4. Run the SVTPARMS batch job to validate that you have properly configured your parmlib options. Correct any parmlib configuration problems before continuing.

5. If your TCP/IP Stack is active, restart SVTS. At startup SVTS should display several messages indicating that CA Vtape P2P is active, for example:

   ```
   SVT1TP208I RC=0,0,P2P@MCSX,EMCS Console SVT7XE61 activated
   SVT1TP000I Console routed to LOCAL
   SVT1TP501I RC=0,0,P2P@LSTN,Host XXI161ME, 41.202.65.61
   SVT1TP500I RC=0,0,P2P@LSTN,Listening on 0.0.0.0:11001
   ```

   The first SVT1TP500I message indicates the default IP address of the TCP/IP stack *(41.202.65.61 in this example)*. The second message indicates that the P2P Listener has established a TCP/IP socket and is waiting for remote connections on the given IP address *(0.0.0.0)* and port *(11001)*.

   Note: The default ListenOnIPAddress attribute in VTP2POPT is 0.0.0.0. This allows the P2P Listener to accept connections from any IP address connected to the TCP/IP stack.

6. Issue a SVTS D P (*display parmlib*) command. This will record the parmlib settings on the console log and show both the CA Vtape P2P options and remote systems as defined by parmlib, for example:

```
PeerToPeerOptions initialized from Parmlib member: VTP2POPT
   ProtocolFamily................. IPV4
   TCPName........................ TCPIP61
   DsnameSYSTCP................... VTAM.TCPIP.TCPIP.DATA
   ListenOnPort................... 11001
   ListenOnIPAddress.............. $NULL
   MaxClients..................... 4
   IdleTaskTimeout................ 30
   EMCSTerminalPrefix............. SVT, EMCS CN=SVT1XX61
PeerToPeerRemotes initialized from Parmlib member: VTP2PRMT
   LCL1 LocalHost…..... 11001 127.0.0.1
```

7. Issue a SVTS D R (*display remotes*) command. This command will show the connection status of any of the defined remote systems, for example:

```
SVT1 D R
SVT1TP403I Display Remote Statistics
Rmt  Status       AsOf   #Receive     #MB Files #Transmit     #MB Files
---- ------------ ----- --------- ------- ----- --------- ------- -----
LCL1 Active       15:08         5       1     0         5       1     0
---------------------- --------- ------- ----- --------- ------- -----
1 Remote System(s)              5       1     0         5       1     0
```

   In the example above note that it shows that LCL1 is active and that we have been able to send and receive data from it.

8. You should now be able to issue remote SVTS commands and effectively test the ability of CA Vtape P2P to communicate with itself as the localhost remote system.

   Note: One of the features of CA Vtape P2P is that it allows you to direct any normal SVTS commands to a connected remote system. This is accomplished by appending the console suffix to the SVTS console command prefix using either a period or a slash character. For example:

   SVT*n* D H is the normal SVTS display help command. To route that command for execution to the localhost remote system simply append its console suffix LCL1, that is, SVT1.LCL1 D H or SVT1/LCL1 D H

   Try a simple command, such as: SVT1.LCL1 D S (*display status*) or SVT1/LCL1 D H (*display help*).

   SVTS operator commands entered in this manner will be transmitted by the *local system* to the *remote system* for execution. SVTS console output displayed by the *remote system command* will then be transmitted back to the *local system*. There, the console suffix will prefix the remote console output to distinguish its source. For example:

```
SVT1.LCL1 D S
SVT1TP204I Command for LCL1, LocalHost scheduled
LCL1X0224I Subsystem Status Display
     Field               Current State
     ------------------- -------------------
     Release             r11.5
     Subsystem           Active
     …
     Recall Order        Primary
     Console             Local
LCL1X0100I Command Complete
```

   Note: The console suffix 'LCL1' was also used as a prefix for messages displayed from that *remote system*. This allows you to determine from which remote subsystem that console output was derived.

You have successfully configure a small CA Vtape P2P parmlib configuration designed to test TCP/IP. In this case the remote subsystem was simply the loopback address.

Set up a SVTS Subsystem in a Separate CA Vtape Complex

In order to create remote Virtual Volumes you need at least two SVTS subsystems which do not share the same cache or control data sets. If necessary, these subsystems can run on the same LPAR but must use different high level qualifiers.

If you do not have one already you can create a separate SVTS subsystem by following the normal CA Vtape installation procedures.

Once you have a separate SVTS subsystem, you can configure CA Vtape P2P for the additional SVTS subsystem in the same manner as defined in the section Set Up a Simple P2P Configuration to Test TCP/IP Connectivity (see page 237). Assuming that both of these subsystems can then communicate through TCP/IP the next step is to define these two subsystems as *remote systems* to each other.

Note: In the following example the first SVTS subsystem is defined as SVT1 and the second subsystem is SVT2.

To define two subsystems as *remote systems* to each other

1. Modify the VTP2PRMT members of each SVTS subsystem so that they can communicate with each other.

   In the VTP2PRMT of the first subsystem place a remote system definition which describes the second subsystem.

   Assuming P2P Listener for SVT2 is using port 11002 and that TCP/IP stack uses IP address 41.202.65.61, you can modify the VTP2PRMT member of SVT1 to contain a remote definition for SVT2, that is:

```
<PeerToPeerRemotes>
    RemoteSystem=LocalHost
    RemoteSystem=SYS61.SVT2

<LocalHost>
   Port = 1100&SVTS(-1:1)
   IPAddress = 127.0.0.1
   ConsoleSuffix = LCL&SVTS(-1:1)

<SYS61.SVT2>
   Port=11002
   IPAddress=41.202.65.61
   ConsoleSuffix=S261
```

Observe the following:

- The combination of IP address and port for each P2P Listener must be unique. In this example different subsystems running on the same LPAR and TCP stack use the same IP address, however notice that the port numbers used by their respective P2P Listeners is in fact different SVT1 is using port 11001 while SVT2 uses 11002.
- Setting up a unique ConsoleSuffix takes a certain amount of preplanning. This user defined, four character field must uniquely identify the *remote system* and also be meaningful and easy to remember. In our examples we are using a naming convention which consist of S*xNN* where *x* identifies the SVTS subsystem number (that is, from 1 to 8) and *NN* identifies the LPAR where that subsystem is running, which in this case is SYS61.

Then assuming that the P2P Listener for SVT1 uses port 11001 and the TCP/IP stack it uses is at IP address 41.202.65.61, you can modify the VTP2PRMT member of SVT2 to contain a remote definition for SVT1, as follows:

```
<PeerToPeerRemotes>
    RemoteSystem=LocalHost
    RemoteSystem=SYS61.SVT1

<LocalHost>
   Port = 1100&SVTS(-1:1)
   IPAddress = 127.0.0.1
   ConsoleSuffix = LCL&SVTS(-1:1)

<SYS61.SVT1>
  Port=11001
  IPAddress=41.202.65.61
  ConsoleSuffix=S161
```

2. Refresh the remote system definitions of each SVTS subsystem by issuing the operator command SVT*n* REFRESH=REMOTES. This command will cause each subsystem to reload its PeerToPeerRemotes (VTP2PRMT) member.
3. After about 30 seconds issue a SVT1 D R (*display remotes*) command. If the two systems are able to communicate, the display will show the status of that communication, for example:

```
SVT1 D R
SVT1TP403I Display Remote Statistics
Rmt  Status       AsOf   #Receive     #MB Files #Transmit     #MB Files
---- ------------ ----- --------- ------- ----- --------- ------- -----
LCL1 Active       17:06         5       1     0         5       1     0
S261 Active       17:08         5       1     0         5       1     0
---------------------- --------- ------- ----- --------- ------- -----
2 Remote System(s)             10       2     0        10       2     0
SVT1X0100I Command Complete
```

In the example above, S261 (SVT2) shows up as active and is able to communicate with SVT1.

In addition, SVT1 should be able to transmit remote SVTS commands and receive responses from SVT2 by simply using the S161 console suffix, for example:

```
SVT1.S261 D S
SVT1TP204I Command for S261, TSO61.SVT2 scheduled
S261X0224I Subsystem Status Display

     Field               Current State
     ------------------- ----------------
     Release             r11.5
     Subsystem           Active
     …
     Recall Order        Duplex
     Console             Local
S261X0100I Command Complete
```

Try issuing SVT1.S261 D R (*display remotes*). You should then be able to see the status remote connections from the point of view of SVT2.

By following this procedure you have been able to configure two SVTS subsystems for basic P2P communication.

Test Remote Virtual Volumes Access

The next logical step is to test your remote Virtual Volume access. Having systems which can communicate, you can now begin configuring one of these SVTS subsystems for remote Virtual Tape processing. This entails setting up remote drives, Virtual Volume pools, remote groups, and finally remote data filters.

To configure a SVTS subsystem for remote Virtual Tape processing

1. Logically, SVTS treats *local Virtual Tape units* (VTU) and *remote VTU*s as two different device types. *Local VTU*s are assigned for local Virtual Volumes while *remote VTU*s are only assigned for remote Virtual Volumes.

   In order to mount or create both types of tapes, you must define both types of devices in the VTDRIVE parmlib member.

   If you already have a sufficient number of Virtual Devices addresses, you may want to reserve some for use as *local VTU*s and others as *remote VTU*s.

   Note: If you need additional device addresses you can define them using IBM's Hardware Configuration and Definition Dialogs (HCD) as described in Precustomization Planning. If your shop is running JES3, you cannot split your local and remote Virtual Volumes into separate pools. Under JES3 you can only define one volume pool.

   Defining local versus remote VTUs is simply a matter of coding the VTDRIVE parmlib member accordingly. The VTDRIVE member of CCUUPARM library includes several examples of how local and remote virtual devices are defined.

Note: In this example we are only going to define remote Virtual Tape units to one subsystem (SVT2). This is the subsystem that we will refer to as the *local system* later in this document.

Begin by modifying the VTDRIVE member of the *local system* (SVT2) to include at least two remote Virtual Units and at least two local Virtual Tape units. For example:

```
;*************************************
; SVT2 Drive List
;*************************************
 <VirtualDeviceList>
; Local VTUs

  VirtualControlUnit = 1
            Offline = 3500-3501
;Remote VTUs

  VirtualControlUnit = 2
       OfflineRemote = 3510-3511
```

2. Restart the SVT2 subsystem. Modifying the Virtual Device definitions of VTDRIVES requires that we restart the subsystem in order for those definitions to be loaded.
3. Issue the SVT2 D U (*display unit*) command to ensure that the Virtual Devices are properly defined to SVT2.
4. Vary the Virtual Devices both offline and online to ensure that they are reacting properly.

5. CA Vtape P2P allows separate SVTS subsystems utilizing *remote VTUs* to logically share an overlapping subset of Virtual Volser ranges. Using Virtual Volume Pooling we can logically isolate Virtual Volser ranges for use with remote mounts versus those that are used for local mounts.

   In this simple example, we configure the volume pools of SVT1 using its VTPOOLS member:

   Note: When you do this be sure to use volser ranges that are appropriate for testing within your own shop.

```
; SVT1

<VolumePoolDefinitions>
  Pool1=Pool1           ; Local volsers
  Pool4=Pool4           ; Remote volsers

<Pool1>
        Range=100000-100999
<Pool4>
        Range=300000-300999

While the VTPOOLS member used by SVT2 is configured as:
; SVT2

<VolumePoolDefinitions>
  Pool1=Pool1           ; Local volsers
  Pool4=Pool4           ; Remote volsers

<Pool1>
        Range=200000-200999
<Pool4>
        Range=300000-300999
```

   In this example the intent is to use POOL4 as the overlapping volser range used for remote Virtual Volumes.

6. Once the VTPOOLS members are properly defined you can issue a SVTS REFRESH=POOLS on both subsystems. This command will load the new pool definitions onto both subsystems.

   Once the VTPOOLS member has been refreshed you can then confirm that they are properly enabled by issuing the SVTS D P (*display parmlib*) or SVTS D U (*display* unit) operator commands.

   Important! Although these subsystems now have an overlapping range of volsers in POOL4, neither system will assign scratch volumes from that range because there is currently no GROUP definition assigned to POOL4.

7. Modify the VTGROUP member used by SVT2 to establish a *remote group definition*.

   A *remote group definition* identifies the name of the *remote system* with whom to communicate and where the remote Virtual Volume will be created. In addition the group definition must specify the Virtual Volume Pool from which to allocate scratch Virtual Volumes. For this example SVT1 is the remote system with whom to communicate and POOl4 is the Virtual Volume Pool to be used:

```
<SVT2.GROUP71>
       Description                = '&SVTS GROUP71'
       Primary                    = TEMPONLY
       VolumePool                 = Pool4
       RemoteSystemPrimary        = /S161

<SVT2.GROUP72>
       Description                = '&SVTS GROUP72'
       Primary                    = TEMPONLY
       VolumePool                 = Pool4
       RemoteSystemPrimary        = SYS61.SVT1
```

   Note: The RemoteSystemPrimary and RemoteSystemSecondary can either be the name of the *remote system* (that is, SYS61.SVT1) or the remote ConsoleSuffix proceeded with a slash character (that is, /S161).

   One important aspect to remember about remote Virtual Volumes is that the group number assigned by the *local system (SVT2)* is the same group number used by the *remote system (SVT1)* for purposes of Externalization. The point is that the group definitions of these SVTS subsystems may be different allowing the Externalization processes of these systems to behave differently.

   In the example above, the Primary attribute is assigned to TEMPONLY for both GROUP71 and GROUP72. In this case, the *local system* (*SVT2*) will not externalize the Virtual Volume locally. Once they are created they are immediately placed on the freeable queue. Later, if these remote Virtual Volumes are deleted from local DASD cache of SVT2 they can be recalled from cache on the remote system, *SVT1*.

Important! If both the local system and the remote system define Primary=TEMPONLY for the same respective Group, then a condition will exist where neither the local system nor the remote system will externalize the Virtual Volume. As soon as the Virtual Volume is freed from cache the volume is lost. Therefore only use such a combination of TEMPONLY for temporary data sets.

GROUP71 and GROUP72 as defined on the *remote system (SVT1) are* not remote groups. For our example the group definitions used by the r*emote system* will be:

```
<SVT1.GROUP71>
       Description              = '&SVTS GROUP71'
       MB_Threshold             = 13000
       ShortRetention           = 0
       MediumRetention          = 7
       Primary                  = P3590
       BackstoreBlockSize       = 256K
       VolumePool               = Pool1
<SVT1.GROUP72>
       Description              = '&SVTS GROUP72'
       MB_Threshold             = 13000
       ShortRetention           = 0
       MediumRetention          = 7
       Primary                  = P3590
       Duplex                   = P3590
       BackstoreBlockSize       = 256K
       VolumePool               = Pool1
```

As a remote Virtual Volume is created on SVT2 it will be transmitted to SVT1. After it is dismounted, it externalized on SVT1 according to the rules defined above. A primary backup will be created for remote volumes created in GROUP71. Likewise SVT1 will create a primary and duplex backup for those created in GROUP72.

8. Activate the modified VTGROUP definitions of both systems by using the SVT*n* REFRESH=GROUPS command. Then verify that the remote group definition is in effect by issuing a SVTS D P,M (*display parmlib*) command.

9. Modify the VTFILTR member defining a set of *remote data filters* on the *local system (SVT2)*. Create remote data class or remote data sets filters for data sets that you intend to assign to the *remote group definitions* created in step 7 above.

   Note: *Remote groups* can only be referenced by *remote data filters*.

   Also note that *local data filters* take precedence over *remote data filters*. If a scratch mount for the same data set can be selected for processing by both a *local* and a *remote data filter* then that data set will be mounted on a local drive rather than a remote drive.

   A good example of the parmlib attributes describing *remote data filters* can be found in the VTFILTR member of the CCUUPARM library.

A simple example of a set of local and remote DataSetFilters:

```
<DataSetFilters>
       IncludeDC       = &SVTS..IncludeDC
       IncludeDS       = &SVTS..IncludeDS
       ExcludeDS       = &SVTS..ExcludeDS
       IncludeDCRemote = &SVTS..Rmt.IncludeDC
       IncludeDSRemote = &SVTS..Rmt.IncludeDS
       ExcludeDSRemote = &SVTS..Rmt.ExcludeDS
;*******************************************
;* Filters for SYS61.SVT1
;*******************************************
 <SVT1.IncludeDC>
<SVT1.ExcludeDS>
<SVT1.IncludeDS>
       GROUP21 = 'hlq.VTAPE21/'
<SVT1.Rmt.IncludeDC>
<SVT1.Rmt.ExcludeDS>
<SVT1.Rmt.IncludeDS>
;*******************************************
;* Filters for SYS61.SVT2
;*******************************************
 <SVT2.IncludeDC>
<SVT2.ExcludeDS>
<SVT2.IncludeDS>
       GROUP01 = 'hlq.VTAPE01/'
       GROUP11 = 'hlq.VTAPE11/'
<SVT2.Rmt.IncludeDC>
<SVT2.Rmt.ExcludeDS>
<SVT2.Rmt.IncludeDS>
       GROUP71 = 'hlq.VTAPE71/'
       GROUP72 = 'hlq.VTAPE72/'
```

10. Once the appropriate *remote data filters* have been defined, activate them on the *local system (SVT2)* by issuing a SVT*n* REFRESH=FILTERS operator command.

    You can then confirm that the new *remote data filters* are in effect by issuing a SVT*n* D P,L *(display parmlib*) command.

    At this point the systems should be configured to begin testing access to *remote Virtual Volumes* through CA Vtape P2P.

    Note: If both the group definitions and the filter definitions have been changed, then after editing is complete for both, either a REFRESH=GRouP,E or a REFRESH=FiLTers,E command will activate both the changes to the groups and filters in the correct order.

11. The next logical step is to attempt to create a remote Virtual Tape on the active system.

    If you have not done so already vary online the *Virtual Vtape Units* created in step 1 above.

    Next, create JCL for a simple job that will copy a data set to tape. Ensure that the output data set matches the criteria of a remote data filter defined in step 9 above, that is:

    ```
    //STEP1   EXEC PGM=IEBGENER
    //SYSPRINT DD  SYSOUT=*
    //SYSUT1   DD  DISP=SHR,DSN=hlq.data.set
    //SYSUT2   DD  DSN=hlq.VTAPE71.data.set,
     // DCB=(DSORG=PS,RECFM=F,BLKSIZE=64536),
     // UNIT=(3490,,DEFER),DISP=(NEW,DELETE)
     //SYSIN   DD  DUMMY,DCB=BLKSIZE=80
    ```

    Submit the job on the *local system* (*SVT2 in our example*). If the remote data set filters are correctly coded on the *local system* then the output DD should be assigned to a *remote VTU*, a remote virtual volser from POOL4 will be assigned and the mount should complete.

    While the job is running a SVT*n* D U (*display units*) will show the activity of the remote and local drives.

    After the job completes the same Virtual Volume data set should be present on both subsystems. You can check that the Virtual Volume is defined to both systems using the CA Vtape ISPF panels.

By following this procedure you should have been able to configure and write remote Virtual Volumes using two CA Vtape complexes.

CA Vtape P2P Advanced Configurations

To control the behavior of CA Vtape P2P, customize the *Group Definitions Primary* and *ExchangeMetadataOnly*. These attributes influence the way network and remote system resources are utilized. In all situations, scratch mounts cause the remote drive to connect to the remote system during the mount. If the connection is dropped, the remote drive reestablishes a connection to the remote system. After a Virtual Volume has been created, the behavior of the remote drive depends on the Group Definitions. These examples show how to customize the Group Definitions.

Example of how to customize the Group Definitions

On the local system, the Group definition contains:

- Remote Group contains Primary=esoteric
- ExchangeMetadataOnly=Y

On the remote system, the Group definition contains:

- Primary=TEMPONLY.

This attribute combination externalizes Virtual Volumes locally and sends as little information over the network as possible. However, the combination still allows the Virtual Volume to be accessed on the drives local of the remote system. This group customization places the least amount of stress on the remote system. Application data does travels over the network only if drives local on the remote system mount a volume that the remote drives create. This is due to the ExchangeMetadataOnly attribute. Additionally, if a Virtual Volume is mounted for read at the local site, no network connections are attempted. However, once a remote drive updates a Virtual Volume a connection with the remote system is established.

Example of how to customize the Group Definitions

On the local system, the Group definition contains

- Remote Group contains Primary=TEMPONLY
- ExchangeMetadataOnly=N

On the remote system, the Group definition contains:

- Primary=esoteric.

The intent behind this combination of attributes is to leverage a small amount of DASD cache on the local site without needing any physical drives. All permanent copies of the Virtual Volumes are kept on the remote system.

Example of how to customize the Group Definitions

On the local system, the Group definition contains:

- Remote Group contains Primary=esoteric
- ExchangeMetadataOnly=N

On the remote system, the Group definition contains:

- Primary=esoteric

The intent behind this combination of attributes is to maintain a highly fault-tolerant environment.

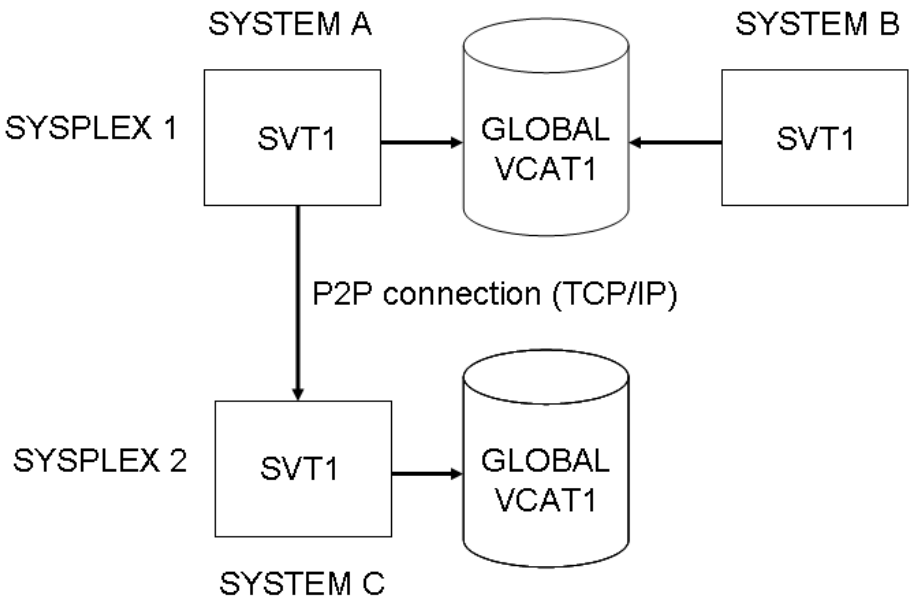
Scratch Processing for Remote Virtual Volumes

Scratch processing in a P2P configuration requires a Virtual Volume to pass new edits to validate the ownership of the Virtual Volume. Remote Scratch volser selection is performed by the Remote System utilizing a round-robin selection process. As Virtual Volumes are created, a unique signature of the remote drives configuration and the remote system configurations are placed into the Virtual Volume record. The remote systems will not allow scratch processing for Virtual Volumes that do not match these signatures. The ownership of the Virtual Volume remains with the remote drives site until the Virtual Volume is scratched by the local system.

During scratch processing on the local site, the signatures of the Virtual Volumes are checked. When the signature passes edit checks the Virtual Volume is immediately scratched locally and a list of candidate volumes is queued for transmission for scratch processing by the remote system. Once the remote site receives the list it also validates the signatures and scratches the Virtual Volumes.

The Scratch Synchronization Job must be executed on a system that has a CA Vtape Subsystem running with a P2P connection to the remote CA Vtape Complex that owns the VOLSERs to be scratched. The SVTS parameter in the VVESYNC step of that job must be set to the subsystem Id of the CA Vtape Subsystem with the P2P connection.

Example scratch processing in a P2P configuration



In the above configuration, the Scratch Synchronization Job must be run on system A with SVTS=SVT1 to scratch the Virtual VOLSERs written on system A and transmitted to system C since both Global VCATs have entries for those VOLSERs. If the Scratch Synchronization Job is run on system B, SVT1 cannot communicate the scratch to system C.

The Scratch Synchronization Job must also be run on system C with SVTS=SVT1 if Virtual Volumes were written on system C regardless of whether or not those Virtuals were transmitted to SVT1 on system A.

Chapter 16: Performance

This chapter describes various aspects of CA Vtape which can increase or decrease CPU usage or memory usage.

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

This section contains the following topics:

Response Time (see page 253)
Exploitation of the zIIP Specialty Processor (see page 254)
CPU Performance Compared to Memory Utilization (see page 256)

Response Time

CA Vtape response time is directly related to available MIPS and the DASD cache volume response times.

If the CPU is constrained, CA Vtape will run slower just like the other software and jobs executing on the LPAR. Jobs attempting to mount Virtual Volumes will experience elongated mount and run-times. Increasing the available MIPS or moving workloads that do not use CA Vtape to other LPARs or installing a zIIP engine can help in this situation.

If the DASD used for the CA Vtape cache has slow response times, CA Vtape reads and writes to that DASD will have slow response times. Jobs attempting to read and write Virtual Volumes will experience elongated run-times. There are a number of ways to address slow DASD response times. The following are some of the most affective ways to address slow DASD response times:

- Defining Parallel Access Volumes (PAV) for the DASD cache volumes.
- Moving the DASD cache volumes to another DASD box that supports faster access for sequential reads and writes.
- Moving some of the DASD cache volumes to another DASD box that is not as busy.
- Increasing the number of channels to the DASD box.

Exploitation of the zIIP Specialty Processor

CA Vtape is designed to exploit IBM zIIP specialty processors when they are installed and enabled on the system where CA Vtape will run and the appropriate parmlib attributes are specified. 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 to run on the zIIP processor.

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 enable to run on a zIIP processor may not be processed on the zIIP for a number of reasons. Refer to the appropriate IBM documentation for further information on the zIIP.

How to 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 SVTSAS.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.

How to Monitor zIIP Utilization

SMF Record 30 (x1E) - Common Address Space Work provides details on zIIP utilization at a specific job level. For the CA Vtape Virtualization Engine, SMF records for the SVTSAS.SVT*n*V*n* address spaces should be extracted.

For work that is run on a general processor, CPU time is accumulated in SMF 30 record field SMF30CPT for task related work or SMF30CPS for SRB work. The CA Vtape Virtualization Engine normally runs in SRB mode; therefore CPU time is mainly held in SMF30CPS.

The SMF 30 record provides for accumulating CPU time for both independent and dependent enclaves of the owning address space. SMF30ENC or SMF30DEP holds CPU time used by the independent or dependent enclaves but only when in the WLM enclave. Time accumulated in SMF30ENC and SMF30DEP is also part of the value in SMF30CPT. CA Vtape uses dependent enclaves.

The SMF30ENC, SMF30DEP, or SMF30CPT fields do not include time spent processing SRBs joined to an enclave that is eligible to run on a zIIP. Instead, time spent on a zIIP is held in fields:

- SMF30_ENCLAVE_TIME_ON_zIIP (independent enclaves) or
- SMF30_DEPENC_TIME_ ON_zIIP (dependent enclaves) and
- SMF30_TIME_ON_zIIP (both independent and dependent enclaves)

CPU Performance Compared to Memory Utilization

CA Vtape uses Data-In-Virtual (DIV) to map dataspaces to its DASD buffer LDSs. The memory occupied by these dataspaces is used to buffer Virtual Volume data before writing it to the LDSs. This provides higher throughput performance for jobs reading or writing data to CA Vtape Virtual Volumes. Unchecked, these dataspace buffers would normally require a large storage footprint; however, CA Vtape minimizes the size of the dataspaces to ensure that they do not burden memory resources. CA Vtape also gives you with the control to prioritize its use of memory as opposed to CPU utilization within your environment.

CA Vtape can be run in one of two modes, STANDARD (see page 256) or ISOLATION (see page 257), which is controlled by the SVT*n* SET CPU console command. Standard Mode is the recommended mode.

STANDARD Mode

In STANDARD mode, CA Vtape maximizes available memory at the expense of some additional CPU overhead. In this mode, CA Vtape actively manages the amount of storage it uses by frequently releasing the dataspace memory pages that have been written to the LDSs.

CA Vtape does this by periodically issuing a DIV RESET command for memory pages that have been previously saved to DASD. By releasing these memory pages, CA Vtape helps maximize available memory for other tasks running within the operating system.

Standard is the recommended and default setting. You can use the Display Status console command to determine the current setting.

Note: The SET CPU command setting is stored in the CA Vtape Local VCAT and remains in effect across IPLs until specifically changed. In a multisystem CA Vtape configuration, it is valid to allow different system images to have different SET CPU settings.

ISOLATION Mode

ISOLATION mode reduces CA Vtape CPU utilization at the expense of slightly higher memory usage. In this mode, CA Vtape allows the operating system to manage the storage footprint. The operating system does not allow the real storage requirements of an address space to grow beyond the parameters you configure for storage isolation. When additional pages are required, the operating system can release pages that have been previously saved. This can result in significant CPU savings because CA Vtape does not need to issue DIV RESET to release saved memory pages.

On heavily loaded online systems with high paging rates and low tape usage, it may be preferable to run CA Vtape in the STANDARD mode. This helps reduce the amount of memory used by CA Vtape. On highly batch-oriented systems, where tape throughput is critical to overall system throughput, consider using ISOLATION mode to reduce CPU cycles needed to manage memory resources.

Note: To use ISOLATION mode, you must correctly configure Storage Isolation within your operating system. To determine the optimum settings for Storage Isolation within your environment, see the chapter "Storage Isolation" in the IBM *OS/390 Initialization and Tuning Guide* and consult with your systems programmer.

Index

A

allocate
  non-SMP/E data sets, VTA08NON • 25
  SVTJCL, VTA08NON • 25
allocation
  DASD allocation control products • 40
  logic, CA MIM/IBM GRS support • 217
AllowBTEncryptionForVVE, DynamicOptions attribute • 116
AllowConcurrentVVEReadAccess attribute • 104
analyzing, how much data • 33
APF authorizations • 19
attribute • 167
attribute feature and location, parmlib • 91
attribute values, parmlib • 83
attributes related by feature, parmlib • 96
audience • 11
AutomatedSubgroups, GroupSection attribute • 143
automatic command execution • 89
automatic Recall source switching • 216
automatic source switching • 216
automation
  subgroup • 207
  subgroup adjustment, tape management system • 77
  subgroup, resetting and overriding • 209
AutoMedia (Zara), stacked or multiple-file tapes • 77

B

backing up
  BSDS • 188
  Global VCAT • 188
Backstore
  default group and subgroup • 206
  EXPORT, implementing • 212
  externalization throughput • 212
  management • 203
  processing, releasing physical tapes • 213
Backstore Engine • 203
  automatic recall source switching • 216
  backstore engine, starting, stopping, restarting • 203
  cache shortage • 215
  configuring the Backstore Engine in Primary and Failover roles • 211
  MAXDRIVES setting • 215
Backstore Tape
  capacity and usage • 204
  drives, maximum • 214
  releasing from backstore processing • 204
Backstore Tapes, contention scenarios • 219
Backstore, USS Backstore • 220
BackstoreBlockSize, GroupSection attribute • 143
BackstorePriority, GroupSection attribute • 143
BackstoreRetryCount, DynamicOptions attribute • 116
BackstoreTimeoutValue, GroupSection attribute • 143
BrightStorEncryption, GroupSection attribute • 143
BrightStorEncryptionDC, GroupSection attribute • 143
BSDS
  backing up • 188
  determining size • 31
  recovering • 189
BypassOfflinePhysicalDevices, StartupOptions attribute • 104
BypassOfflinePhysicalDevices=N • 218
BypassOfflinePhysicalDevices=Y • 218
BypassRLLCompression, DynamicOptions attribute • 116
BypassUCBChecking, StartupOptions attribute • 104

C

CA 1
  stacked or multiple-file tapes • 75
  Tape Management System, synchronizing • 68
CA Allocate
  HSC exits • 29
  HSC tapereq parameters • 29
  Offline=, VirtualDriveList attribute • 139
CA Disk
  enabling concurrent read access to Virtual Volumes • 201
  recovering the BSDS • 189
  simultaneous read access for ARCHVOLS on any LPAR • 201
CA MIM/IBM GRS support, allocation logic • 217

CA OPS/MVS
enable event notification • 30
health check state management • 30
integration • 29
OfflineRemote=, VirtualDriveList attribute • 139
system state management (SSM) • 30
CA Tape Encryption interface, Virtual Volume • 201
CA TLMS
retention types • 76
stacked or multiple-file tapes • 76
tape management • 70
CA Vtape P2P option • 233
cache
DefaultGroup=, DynamicOptions attribute • 116
determine size • 32
management strategy • 34
Online=, VirtualDriveList attribute • 139
static cache DASD volumes for • 42
cache shortage • 215
CacheAutoHoldLowThreshold, DynamicOptions attribute • 116
CacheDefaultDataClass, StartupOptions attribute • 104
CacheResidenceHours, GroupSection attribute • 143
CacheWarningThreshold, DynamicOptions attribute • 116
CANCEL, (VOLUME_ENQ) detecting volume contention • 191
capacity and usage, physical tape • 204
CatalogManagedDate, DynamicOptions attribute • 116
CatalogManagedSubgroup, GroupSection attribute • 143
channel paths, virtual device • 197
CHPIDs
ChpidDeviceList= • 139
per Virtual Device • 199
virtual device channel paths • 197
command execution, automatic • 89
concurrent read access to virtual volumes created by CA Disk • 201
configuration process • 11
configure, multisystem • 14
configuring the Backstore Engine in Primary and Failover roles • 211
ConsoleCommandTimeout, DynamicOptions attribute • 116
ConsoleSuffix attribute • 164
contention over Backstore Tapes • 219
control
control, data sets • 55
DASD allocation control products • 19
files availability • 188
Control-T
(BMC), scratch synchronization • 70
Control-T, stacked or multiple-file tapes • 76
Copy of subgroup automation
subgroup automation, overview • 207
subgroup automation, resetting and overriding • 209
copying data sets while in use • 193
CPU performance • 256
ISOLATION mode • 257
STANDARD mode • 256
zIIP exploitation • 254
customization, ISPF process • 45
customizing PROCs • 180

D

D(efine), ISPF customization panels • 46
DASD
allocation control products • 19
DASD allocation control products • 40
volumes, determining number of static cache • 42
data class filters, how P2P works • 234
data set
allocate non-SMP/E, job VTA08NON • 25
buffer, cache management strategy • 34
control • 55
copying data sets while in use • 193
dynamic and static DASD buffer data sets • 34
filters • 166
filters, remote how P2P works • 234
name filtering • 60
support for data set stacking • 25
using both SMS data class constructs and data set name filtering • 60
data set filters, how P2P works • 234
Delete, JCL • 47
Description, GroupSection attribute • 143
determining static cache DASD volumes • 42
DFSMS
data class constructs • 59
HSC exits • 29
HSC tapereq parameters • 29
managed cache, static cache mode • 43

using both SMS data class constructs and data set name filtering • 60
VMA, disk buffer sizing with • 32
DFSMSrmm, JCL • 70
DFSMSrmm, RMM stacked or multiple-file tapes • 76
DsnameBSDS1, StartupOptions attribute • 104
DsnameGlobalVCAT, StartupOptions attribute • 104
DsnameLocalVCAT, StartupOptions attribute • 104
DsnameSYSTCPD, attribute • 159
Duplex, GroupSection attribute • 143
dynamic
dynamic and static DASD buffer data sets • 34
path support • 199
Dynamic Cache
DFMS Data Class • 36
ISPF customization steps • 47
management set up • 36
SMS ACS Routines • 38
SMS Management Class • 37
SMS Storage Class • 37
SMS Storage Group • 38
steps for establishing dynamic cache management • 35
Virtual Volume sizing • 40
DynamicOptions
list of the attributes • 116
ParmlibDirectory attribute • 101
DynamicOptions attribute • 116

E

EMCSTerminalPrefix, attribute • 159
encryption, CA Tape Encryption interface • 201
enqueues
SVTRCYCL • 17
SVTSX • 17
enqueues and reserves • 15
SVTS Reserve • 15
SVTSX and SVTRCYCL • 17
esoterics
defining Virtual Devices using HCD • 18
using, tape mount redirection • 61
ExchangeMetadataOnly, GroupSection attribute • 143
ExcludeDatasets
DatasetFilters attribute • 167
Group attribute • 170
ExcludeDataSetsRemote, DatasetFilters attribute • 167
ExcludeDS, DatasetFilters attribute • 167
ExcludeDSN, DatasetFilters attribute • 167
ExcludeDSNRemote, DatasetFilters attribute • 167
ExcludeDSRemote, DatasetFilters attribute • 167
EXEC.SVTJCL • 182
export tapes • 212
Export, GroupSection attribute • 143
external logger
defining • 19
System Logger Log Stream • 21
externalization
configuring the Backstore Engine in Primary and Failover roles • 211
server subgroup queue management • 205
throughput, backstore • 212

F

Failover, configuring the Backstore Engine in Primary and Failover roles • 211
filter
data class filters, how P2P works • 234
data set name • 60
processing • 64
tape mount intercept filters • 62
foreign tape processing • 78
ForeignTapesExpdt, DynamicOptions attribute • 116
FullMaxdrivesEnforcement settings
set to No • 214
set Yes • 215
FullMaxdrivesEnforcement, DynamicOptions attribute • 116

G

generics, using • 61
global VCAT
and BSDS, determining size • 31
backing up • 188
recovering, operational considerations • 189
GlobalReserve, DynamicOptions attribute • 116
group
and subgroup, Backstore • 206
GroupDefinitions
ParmlibDirectory attribute • 101
values and description • 142
GROUPnn
=GroupSectionName, GroupDefinitions attribute • 142, 143
IncludeDataClass attribute • 169

IncludeDatasets attribute • 170
GroupSections • 143

H

HardwareCompressionMethod, DynamicOptions attribute • 116
HardwareCompressionOption
DynamicOptions attribute • 116
GroupSection attribute • 143
health check state management • 30
high-speed open (HSOPEN) option • 190
Host Software Component (HSC) • 26
HSC exits
CA Allocate • 29
SMS • 29
HSC tapereq parameters
CA Allocate • 29
SMS • 29
HSOPEN option • 190

I

IBM
defining Virtual Devices using HCD • 18
GRS/CA MIM support, allocation logic • 217
IBM Resource Management Facility (RMF)
Service Class report (WLMGL), monitoring zIIP utilization • 254
RMM, DFSMSrmm, scratch synchronization • 70
understanding the Volume Mount Analyzer • 32
IdleTaskTimeOut, attribute • 159
IncludeDataClass
DatasetFilters attribute • 167
valid values, feature, and descriptions • 169
IncludeDataClassRemote, DatasetFilters attribute • 167
IncludeDatasets
DatasetFilters attribute • 167
valid values, feature, and descriptions • 170
IncludeDatasetsRemote, DatasetFilters attribute • 167
IncludeDC, DatasetFilters attribute • 167
IncludeDCRemote, DatasetFilters attribute • 167
IncludeDS, DatasetFilters attribute • 167
IncludeDSN, DatasetFilters attribute • 167
IncludeDSNRemote, DatasetFilters attribute • 167
IncludeDSRemote, DatasetFilters attribute • 167
installation
HSC considerations • 26
procedures, customization • 55
intercept, tape mount intercept filters • 62
IOCpuTimeout, DynamicOptions attribute • 116
IPaddress attribute • 164
ISOLATION mode, CPU performance • 257
ISPF
customization panels • 45
customization steps • 47
ISPF customization panels
(D)efine • 46
(R)egenerate • 46
(V)iew • 46

J

JCL
AutoMedia • 70
BMC • 70
CA 1 • 68
CA TLMS • 70
Control-T • 70
DELETE • 47
DFMSrmm • 70
ZARA • 70
JCL setup
VMA consolidation of SMF data • 33
JES3 requirements • 17
jobs
static cache submit SUBLDSA • 182
submit static cache LDSADDxx • 182

L

LDSADDxx static cache jobs, submit • 182
library
LicensedLocalVTD, StartupOptions attribute • 104
LicensedRemoteVTD, StartupOptions attribute • 104
ListenOnIPaddress, attribute • 159
ListenOnPort, attribute • 159
local
define the local VCATs • 56
loadlib • 25
SVTJCL • 25
Virtual Tape Units, with the P2P option • 234
LogCSAsize, StartupOptions attribute • 104
LogDataspaceSize, StartupOptions attribute • 104
LogDetailLevel, DynamicOptions attribute • 116
logger

data, preserving external • 192
defining the log stream • 21
log stream general considerations • 21
the IBM system logger • 20
logical partition, subsystems on the same • 192
LogStream, DynamicOptions attribute • 116

M

management, Backstore • 203
managing Externalization Server subgroup queue • 205
MaxClients, attribute • 159
MAXDRIVES setting • 215
maximum physical tape drives and enforcement • 214
MaximumCompressionCPU=
DynamicOptions attribute • 116
GroupSection attribute • 143
MB_Threshold, GroupSection attribute • 143
MediumRetention, GroupSection attribute • 143
member names, parmlib • 84
MessagePrefix, StartupOptions attribute • 104
MIHTimeoutValue, StartupOptions attribute • 104
MinimumCompressionRate=
DynamicOptions attribute • 116
GroupSection attribute • 143
modeling
running • 32
modifying the PROCs
procedure • 180
region size • 180
mount point directory sections • 175
mount, tape mount intercept filters • 62
MountRejectThreshold, DynamicOptions attribute • 116
multiple subsystems
on the same logical partition • 192
renaming the SVTS and SVTSAS PROCs • 179
multisystem
configurations • 14
planning • 13

N

NeverExpireDate, DynamicOptions attribute • 116
NeverExpireSubgroup, GroupSection attribute • 143
non-SMS managed cache, static cache mode • 44

O

OffsiteBackstoreCopy, GroupSection attribute • 143
OMVS segment, P2P security considerations • 235
OnlineRemote=, VirtualDriveList attribute • 139
overriding and resetting subgroup automation • 209

P

P2P
advanced configurations • 250
automatic Recall source switching • 216
CA Vtape P2P option • 233
configuring for CA Vtape subsystems • 236
examples how to customize the Group Definitions • 250
how it works • 234
local Virtual Tape Units • 234
P2P, subpooling • 24
parmlib VTP2POPT • 159
parmlib VTP2PRMT • 163
PeerToPeerOptions attributes • 159
PeerToPeerRemotes attributes • 163
remote data class filters, how it works • 234
remote data set filters, how it works • 234
remote group definition, how it works • 234
remote systems, how it works • 234
remote Virtual Tape Units • 234
scratch mount requested by a remote virtual tape unit • 234
scratch processing for remote Virtual Volumes • 251
security considerations • 235
setting up a simple P2P configuration to test TCP/IP • 237
setting up a SVTS subsystem • 241
testing remote Virtual Volumes access • 243
parameters
BypassOfflinePhysicalDevices allocation logic, CA MIM or IBM GRS • 217
HSC • 28
NNLBDRV HSC • 28
parmlib member syntax • 82
SubAddressSpaceName • 203
TAPEREQ • 28
PARMDIR= keyword • 101
parmlib • 82
attribute feature and location • 91
attribute values • 83
attributes related by feature • 96

feature • 91
member names • 84
setting it up • 85
support • 82
symbolic substitution • 83
syntax • 82
syntax verification • 88
VTDRIVE • 139
VTFILTR • 166
VTGROUP • 142
VTP2POPT • 159
VTP2PRMT • 163
VTPARMS • 101
VTPCMDS • 175
VTPOOLS • 171
VTSCMDS • 174
VUSSMNTS • 175
parmlib member • 171
ParmlibDirectory • 101
ParmlibDirectory attribute • 101
ParmlibDirectory attribute, options • 101
ParmlibDirectory attribute • 101
partition, subsystems on the same logical • 192
path support, dynamic • 199
PeerToPeerOptions attributes • 159, 163
PercentRunOnZIIP, DynamicOptions attribute • 116
physical tape (see also Backstore Tape)
capacity and usage • 204
physical tape drives, maximum • 214
releasing from backstore processing • 213
physical, contention over • 219
pool procesing
scratch • 71
with P2P option • 73
with Virtual Volume pooling • 71
without Virtual Volume pooling • 71
Pooln= attribute • 171
Port= attribute • 164
preserving external logger data • 192
Primary, GroupSection attribute • 143
processing
CPU performance • 256
how to monitor zIIP utilization • 255
how to monitor zIIP utilization for a service class • 254
ISOLATION mode • 257
STANDARD mode • 256
zIIP exploitation • 254
PROCs
modifying • 180
renaming the SVTS and SVTSAS PROCs • 179
ProtocolFamily, attribute • 159

Q

queue management, Externalization Servers subgroup • 205

R

R(egenerate), ISPF customization panels • 46
read, concurrent read access • 191
RealStorageSafetyThreshold, DynamicOptions attribute • 116
recall
configuring the Backstore Engine in Primary and Failover roles • 211
RecallAttemptsThreshold, GroupSection attribute • 143
RecallNotificationEvent, DynamicOptions attribute • 116
RecallServer, DynamicOptions attribute • 116
RecallServerTimeout, DynamicOptions attribute • 116
recovering
BSDS • 189
Global VCAT, operational considerations • 189
Local VCAT • 190
RecvTimeOut attribute • 164
redirection, tape mount • 59
releasing physical tapes from backstore processing • 213
remote
group definition, how P2P works • 234
remote data set and class filters, how P2P works • 234
scratch processing for remote Virtual Volumes, P2P • 251
systems, how P2P works • 234
Virtual Tape <PeerToPeerOptions> attributes • 159
Virtual Tape <PeerToPeerRemotes> attributes • 163
RemoteSystem attribute • 163
RemoteSystemPrimary, GroupSection attribute • 143
RemoteSystemSecondary, GroupSection attribute • 143
renaming the SVTS and SVTSAS PROCs • 179

report
IBM Resource Management Facility (RMF)
Service Class report (WLMGL), monitoring zIIP utilization • 254
zIIP utilization • 254
requirements
JES3 • 17
multisystem • 14
multisystem configurations • 14
reserves
and enqueues • 15
SVTS Reserve • 15
resetting and overriding subgroup automation • 209
response time • 253
RMM
scratch synchronization job • 70
stacked or multiple-file tapes • 76
RoutingCodeCritical, StartupOptions attribute • 104
RoutingCodeNormal, StartupOptions attribute • 104
running CA Vtape modeling • 32

S

scratch
mounts, with the P2P option • 234
processing for remote Virtual Volumes, P2P • 251
tape synchronization • 67
ScratchVolumesThreshold, DynamicOptions attribute • 116
server queue management, Externalization Server subgroup • 205
servers, Failover Server • 211
Service Class, monitor zIIP utilization for • 254
Service Request Block (SRB) • 254
ShortRetention, GroupSection attribute • 143
shutdown, startup and shutdown Commands • 89
ShutdownCommands, ParmlibDirectory attribute • 101
sizing, disk buffer with SMS/VMA • 32
SLSUX02, HSC • 26
SLSUX02J • 26
SLSUX08 HSC • 26
SMF
input record types • 33
Softek Transparent Data Migration Facility (TDMF), copying data sets • 193
SpecialRetentionDate, DynamicOptions attribute • 116
SpecialRetentionSubgroup, GroupSection attribute • 143
specifications, multisystem configurations • 14
stacked or multiple-file tapes
AutoMedia (Zara) • 70
CA 1 • 75
CA TLMS • 76
Control-T (BMC) • 76
DFMSrmm • 76
RMM • 76
support • 75
stacking
data sets on physical tapes • 25
location, Virtual Volume • 78
STANDARD mode CPU performance • 256
start SVTS subsystem • 181
startup command • 89
StartupCommands, ParmlibDirectory attribute • 101
StartupOptions • 104
StartupOptions attribute • 104
static and dynamic DASD buffer data sets • 34
static cache
determining number of static cache DASD volumes • 42
ISPF customization steps • 47
management set up • 43
non-SMS-managed cache • 44
SMS-managed cache • 43
steps for establishing Static Cache Management • 41
Virtual Volume size • 41
STK tape devices, HSC • 26
Storage Management Component (SMC) • 26
Storage Tek, see STK • 26
SubAddressSpaceName, StartupOptions attribute • 104
subgroup and group, Backstore • 206
SUBLDSA job, submit static cache • 182
subpooling • 24
subsystems, setting up a SVTS subsystem for P2P • 241
SVTJCL
EXEC • 182
local library • 25
SVTS
renaming the SVTS and SVTSAS PROCs • 179
Reserve • 15
setting up subsystems for P2P • 241
start subsystem • 181

SVTSAS, renaming the SVTS and SVTSAS PROCs • 179
syntax verification, parmlib • 88
System State Management (SSM) • 30

T

tape
  definning an exlusive range for the Virtual Volumes • 24
  devices, STK, HSC • 26
  mount intercept filters • 62
  mount redirection • 59
  physical, capacity and usage • 204
  physical, releasing from backstore processing • 213
  processing foreign • 78
  scratch tape synchronization • 67
  stacking data sets on physical tapes • 25
  support, stacked or multiple-file • 75
  tape mount intercepts, using esoterics or generics • 61
tape management system
  automatic subgroup adjustment • 77
  AutoMedia (Zara), scratch tape synchronization • 70
  CA 1, scratch tape synchronization • 68
  CA 1, stacked or multiple-file tape support • 75
  CA TLMS, scratch tape synchronization • 70
  considerations • 23
  Control-T (BMC), scratch tape synchronization • 70
  IBM's RMM, scratch tape synchronization • 70
  interface • 77
  other • 71
  sharing the tape management system database • 23
  subpooling • 24
  Virtual Volume expiration date • 77
  Virtual Volume, stacking location • 78
TapeManagementSystem, DynamicOptions attribute • 116
Tasklib, StartupOptions attribute • 104
TCP/IP
  P2P security considerations • 235
  P2P, TCP/IP under UNIX System Services • 235
  setting up a simple P2P configuration to test TCP/IP • 237
TCPName, attribute • 159
TMSVirtualToPhysicalReport, GroupSection attribute • 143

U

UCB, define virtual using HCD • 18
unit, replacement • 29
UNIX System Services, P2P security considerations • 235
UpdateRMFStatistics, StartupOptions attribute • 104
usage and capacity, physical tape • 204
USLSUX02, HSC • 26
USS Backstore • 220
  configure the USS Backstore file system • 224
  customizing mount points • 228
  exploiting NFS server replication • 228
  file system reconfiguration • 230
  how it works • 221
  NFS performance • 229
  of a Virtual Volume • 220
  security considerations • 223
  USS file systems • 222
  USS path setup • 222
utilization, how to monitor for a service class • 254

V

V(iew), ISPF customization panels • 46
verifying
  the system is operational • 182
  the virtual devices are operational • 182
virtual
  defining Virtual Devices using HCD • 18
Virtual Control Units • 196
Virtual Devices
  channel paths CHPIDs • 197
  define primary and alternate using IBM's HCD • 18
  engine • 195
  verifying they are operational • 182
  virtual device channel paths • 197
Virtual Tape Units, local and remote, with the P2P option • 234
Virtual Volume
  CA Tape Encryption interface • 201
  compression • 200
  compression, parmlib feature • 200
  definning exclusive tape range • 24
  expiration date, tape management system • 77

READ by multiple virtual drives at the same time • 201
READ by multiple virtual drives at the same time, contention and concurrent • 191
remote, how P2P works • 234
scratch processing for remote Virtual Volumes, P2P • 251
sharing the tape management system database • 23
size, dynamic cache • 40
size, static cache • 41
stacking location • 78
testing remote Virtual Volumes access for P2P • 243
VirtualControlUnit, VirtualDriveList attribute • 139
VirtualDriveList • 139
VMA
JCL setup, consolidation of SMF data • 33
understanding the IBM VMA • 32
VolserRange attribute • 171
volume
contention and concurrent read access • 191
determining number of static DASD volumes for cache • 42
virtual compression • 200
virtual compression, parmlib feature • 200
Volume Pool, subpooling • 24
Volume Pool Definitions Section • 171
Volume Pool, subpool planning • 24
VOLUME_ENQ, detecting volume contention • 191
VolumePool=, GroupSection attribute • 143
VSLSUX02, HSC • 26
VTA08NON, allocate non-SMP/E data sets • 25
VTDRIVE, parmlib member • 139
VTFILTR, parmlib member • 166
VTGROUP, parmlib member • 142
VTP2POPT Parmlib Member • 159
VTP2POPT, parmlib member • 159
VTP2PRMT, parmlib member • 163
VTPARMS, parmlib member • 101
VTPCMDS, parmlib member • 175
VTSCMDS, parmlib member • 174
VTU, local and remote, with the P2P option • 234
VUSSMNTS • 175

W

WAIT, (VOLUME_ENQ) detecting volume contention • 191
Work Load Manager (WLM) • 19
WTOR, (VOLUME_ENQ) detecting volume contention • 191

Z

zIIP
exploitation • 254
PercentRunOnZIIP= • 254
utilization, how to monitor • 255
utilization, how to monitor for a service class • 254
zIIPExploitation= • 254
zIIPExploitation, StartupOptions attribute • 104