Virtual Tape Control Software

Offsite Vault Disaster Recovery Guide

Version 2.0

docs.sun update only

June 2010, Revision H

Submit comments about this document by clicking the Feedback [+] link at: http://docs.sun.com

About this Book

Oracle's StorageTek VTCS Offsite Vault Disaster Recovery (DR) Feature is an optional, orderable feature of VTCS. **Note that** ExLM, which is used to automate the offsite vaulting process, is also required.

Audience

This guide is for StorageTek or customer personnel who are responsible for using VTCS, the VSM Vault Utilities, and ExLM to create Disaster Recovery (DR) plans and methods for specific customer implementations.

Prerequisites

To perform the tasks described in this guide, you should already understand the following:

- MVS or OS/390 operating system
- JES2 or JES3
- System Management Facility (SMF)
- System Modification Program Extended (SMP/E)
- Nearline Control Solution (NCS)
- VTCS and VSM

About the Software

This guide applies to VTCS Offsite Vault Disaster Recovery (DR) Feature Version 2.0, which runs on VTCS, NCS, and ExLM 4.0.0 and above.

How this Guide is Organized

This guide contains the following sections that describe the offsite vault DR solutions that use VSM and the StorageTek Vault Utilities:

- “What is the VTCS Offsite Vault Disaster Recovery Feature?”, which gives you an overview of the feature.
- “Planning for VTCS Offsite Vaulting”, where you plan out your implementation.
- “Installing the VSM Vault Utilities” on page 9 where we install the utilities you’ll use for the procedures in the following sections.
- “Offsite Vaulting - The Pickup Truck Access Method for CA-1”. “PTAM” stands for “Pickup Truck Access Method”, and its meaning is that after we put your valuable data on tape (square or otherwise), we get a pickup truck to cart the tapes off to the disaster proof vault on the other side of town. This is a very common DR strategy, and VSM is a natural for it because of the power of the `EXPORT` and `IMPORT` facilities and the accompanying Manifest File that contains the VTV and MVC metadata.

  Note that there are actually two flavors of PTAM. The first is for customers who want to manage vaulting and rotation of MVCs (and the VTVs they contain) by data set retention periods.

  Also within this section, and this is still PTAM, we cover data sets **without** retention periods. Instead, we’re using GDGs for MVS Catalog Controlled data sets to manage the aforementioned vaulting and rotation.

The additional sub-flavors for both types of data sets are that we have a TMS and StorageTek utilities for assistance. The utilities we use for this process currently support:

- CA-1
- CA-TLMS
- Control-M/Tape
- DFSMSrmm
- Zara (Zara support requires PTF l1H12ie)

- “Offsite Vaulting - The Pickup Truck Access Method for CA-TLMS”.
- “Offsite Vaulting - The Pickup Truck Access Method for Control-M/Tape”
- “Offsite Vaulting - The Pickup Truck Access Method for DFSMSrmm”
- “Offsite Vaulting - The Pickup Truck Access Method for Zara”.
- “The VSM Vault Utilities”. This is reference information about the `SWDIMS01` and `SWDIMS01` utilities, which we’ll get to know real well in either of the PTAM scenarios.
- We also give you a flow chart of how this all works in “A Sample Offsite Vaulting Work Flow”...
- ...and some helpful fill-in-the-blanks tables in “Offsite Vaulting Configuration Record” where you can record your planning decisions.

What's New in This Guide?

Revision H

The Revision H of this guide contains technical updates and corrections.

Revision G

The Revision G of this guide contains the updates described in Table 1.

Table 1. Updates, Revision G

This SPE...	...is described in...	...via PTF...
added message SWD1960E	"Messages" on page 150	L1H1481

Revision F

The Revision F of this guide contains the updates described in Table 2.

Table 2. Updates, Revision F

This SPE...	...is described in...	...via PTF...
removal of requirement for time stamp file	"SWDTMS01 Utility" on page 144	L1H1428 (SWD6000)
deleted messages: SWD1011W SWD1102I SWD1103I SWD1104I SWD1906E SWD1907E SWD1908E, added messages: SWD1052W SWD1959E changed messages: SWD1050W SWD1112I	"Messages" on page 150	
Addition of STORCLAS CTVOLSNUM parameter for Control-M/Tape	"STORCLAS" on page 169	L1H144S (SWD6000)

Revision E

The Revision E of this guide contains technical updates and corrections.

Revision D

The Revision D of this guide contains the updates described in Table 3.

Table 3. Updates, Revision D

This SPE...	...is described in...	...via PTF...
addition of DD card	no longer applicable with PTF L1H1428	L1H13F0 (SWD6000)
deleted message SWD101W, added message SWD1011W	“Messages” on page 150	

Revision C

The Revision C of this guide contains technical updates and corrections.

Revision B

The Revision B of this guide contains the updates described in Table 5.

Table 4. Updates, Revision B

This SPE...	...is described in...	...via PTF...
addition of PreserveMaximumVTCSReturnCode parameter	“STORCLAS” on page 169	L1H12N (SWD6000)

Revision A

The Revision A of this guide contains the updates described in Table 5.

Table 5. Updates, Revision A

This SPE...	...is described in...	...via PTF...
Zara Support	“Offsite Vaulting - The Pickup Truck Access Method for Zara” on page 119	l1H12ie (SWD6000)

Contents

What is the VTCS Offsite Vault Disaster Recovery Feature?

The VTCS Offsite Vault Disaster Recovery (DR) Feature is an optional, orderable feature of VTCS. **Note that** ExLM, which is used to automate the offsite vaulting process, is also required. You can use VSM2s, VSM3s, or VSM4s.

The VTCS Offsite Vault DR Feature consists of the VSM Vault Utilities and the *VTCS Offsite Vault Disaster Recovery Guide* (this book). This guide tells how to implement offsite vaulting using the "Pickup Truck Access Method" (PTAM) and recover from a disaster (if necessary) by doing the following:

- Setting up the system to withstand a failure and use a Tape Management System (TMS), ExLM, VTCS, and the Vault Utilities to vault critical data and reuse MVCs, as described in "How Does the VTCS Offsite Vault DR Feature Work?" on page 2.
- Continuing business operations if a disaster occurs.
- Resuming normal operations once the Production Site is up and running again.

PTAM is a common DR strategy where data sets requiring safeguarding are migrated to MVCs that are vaulted offsite. VSM adds value to PTAM because of VSM's unique ability to effectively stack VTVs on MVCs. In addition, the VSM Vault Utilities, plus the power of the `IMPORT` and `EXPORT` facilities and the accompanying Manifest File, help you to easily manage your DR vaulting. The VTCS Offsite Vault DR Feature supports the following as TMSs:

- CA-1
- CA-TLMS
- Control-M/Tape
- DFSMSrmm
- Zara (Zara support requires PTF L1H12IE)

What Are the Benefits of the VTCS Offsite Vault DR Feature?

The VTCS Offsite Vault DR Feature provides enhanced data availability and business protection capabilities, and does so using the media optimization advantages of VSM.The early numbers we have from customers are a 50:1 reduction in number of tapes being sent offsite each week and a corresponding reduction in the number of enters and ejects. This resource savings is largely due to moving from Timberline technology to VSM and 9840s. This all translates to reduced cost of slots in the offsite vaults and reduced operator time for handling tapes.

How Does the VTCS Offsite Vault DR Feature Work?

The VSM Vault Utilities, working with VTCS and ExLM, automates sending MVCs offsite, defragmenting offsite MVCs, and then bringing them back onsite for reuse. These utilities support both data sets with retention periods and catalog-controlled data sets. For catalog-controlled data sets, the fragmentation of offsite MVCs occurs because VTVs with different expiration dates reside on the same MVC. The VTCS `EXPORT` and `IMPORT` utilities/commands enable the MVCs for offsite vaulting and return.

For example, using CA-1 as the TMS, the VSM Vault Utilities, working with VTCS and ExLM, let you do the following:

1. Set up to migrate critical data sets on VTVs to MVCs. For data sets with retention periods, VTVs in each retention period are grouped on a common set of MVCs by Storage Class. For catalog-controlled data sets, the VTVs are sent to MVCs in an "offsite" Storage Class.
2. Run CA-1 VMS batch jobs to assign vault codes and rotation to the MVCs
3. Write `TAPEREQs` to migrate critical data sets to MVCs and use ExLM to eject MVCs (by vault code) for offsite vaulting.
4. To recycle MVCs with catalog controlled data sets **or** sites with catalog controlled data sets **and** data sets with retention periods:
    a. Run an ExLM custom volume report to detect MVCs with less than a specified percentage used.
    b. Run the `SWDIMS02` utility to process the ExLM report.
    c. Run the `SWSADMIN` program to "logically drain" the offsite MVCs. The onsite copy of the MVC is actually drained, and the offsite MVC loses its Storage Class and the readonly status is reset off, making it eligible for return and reuse.

    This process **feeds into** the process for recycling MVCs with data sets with retention periods in Step 5.
5. To recycle MVCs with data sets with retention periods:
    a. Run an ExLM custom volume report to detect MVCs with no current VTVs.
    b. Run the `SWDIMS02` utility and the `SWSADMIN` program to process the ExLM report by marking the MVCs expired in the TMC and resetting their readonly status off, making them eligible for return and reuse.
    c. Run CA-1 VMS batch jobs to create a list of MVCs to be returned from the offsite vault for reuse.
    d. Use VMS picklists, identify any MVCs ejected in Step 3. These MVCs are the new batch of MVCs to be vaulted offsite.

Planning for VTCS Offsite Vaulting

This section provides planning information for VTCS Offsite vaulting. You can record your sites values in “Offsite Vaulting Configuration Record” on page 177.

Vault Utilities MVC Usage Guidelines

The VTCS Offsite Vault DR solution lets you achieve your primary objective, which is to safeguard your valuable data. Your secondary objective is to optimize MVC usage, and we provide guidelines for doing so in this section.

So your first question is “How many MVCs will the vault utilities use (per run)?” The *minimum* number of MVCs used is ((x * y) + z) where:

x is the number of Storage Classes used.

y is the number of VTSSs that contain the DR VTVs.

z is the...well, that takes some New Math as follows:

1. Calculate how much data is in each Storage Class (physical size) on each VTSS. This is extra work, but including this calculation improves the accuracy of your answer.
2. For each Storage Class on each VTSS divide the amount of data by the capacity of the MVCs used.
3. For any answer over 1 subtract 1 from the answer then round up to the next integer and add to z.

For example, I have 7 Storage Classes and 4 DR VTSSs, and I calculate *z* as follows:

1. I use 20 Gb MVCs.
2. 5 of the Storage Classes produce less than 20 Gb physical on each VTSS, so they can be ignored.
3. 2 of the Storage Classes each produce 25 GB physical on 2 of the VTSSs.

So I calculate:

z = 2 Storage Classes x 2 VTSSs x (25 / 20 = 1.25 - 1 =.25 rounded up = 1) = 4

And finally:

((x * y) + z) = ((7 * 4) + 4) = 32 MVCs used at a minimum.

Moving right along, your next questions are “What will cause more than the minimum number to be used? What diagnostics can I use to find out what is going on? What are the consequences of taking different choices?” Well, consider the following:

1. You *reduce* the minimum number of MVCs by reducing the number of VTSSs on which the DR data is located. However, this increases the management effort and impacts availability.
2. You *reduce* the minimum number of MVCs by reducing the number of Storage Classes on which the DR data is located. However, this can reduce the effectiveness of the defragmentation process, and cause more MVCs to be retained offsite than is necessary. The “right” number of Storage Classes is a highly tunable item, and is very workload dependent.
3. You *increase* the minimum number of MVCs by spreading the creation/migration of the DR VTVs over a long period of time when the system is busy. VTCS will try to reselect an existing dismounted DR Storage Class MVC with space, but may not always be able to do this, since it must obey its “maximize throughput” algorithms.
4. You *increase* the minimum number of MVCs by allowing automatic migration to run during the DR VTV creation/migration period. If the system is very busy, and immediate migration VTVs have been waiting on the queue, the migration scheduler will increase the number of subtasks serving the immediate migration Storage Class. This spreads the DR VTVs over multiple MVCs. Throughput is increased, but MVC use is decreased.

   An excellent diagnostic technique is to issue the `VT D/QU MIGRATE DET` command every 10 minutes (or increment over 5 minutes but less than the `RETAIN` interval) during the migration/creation period, and determine which Storage Classes are being served. If a DR Storage Class has more than one thread active, the migration scheduler is spreading activity to improve throughput, which increases MVC usage. The solution is to prevent automatic migration from running during the DR VTV creation/migration period by demand migrating down to a threshold before this period.
5. The same situation obtains for automatic space reclamation, which stresses the system, and forces the migration scheduler to make choices that favor throughput over MVC usage. Instead, do demand space reclamation before the DR migration/creation period.
6. What is `MAXMIG`? What is `MINMIG`? Both of these values affect MVC usage. If `MAXMIG` is high, it favors throughput over MVC usage. If `MINMIG` is greater than 1, it favors throughput over MVC usage.

7. What are the expiration patterns of the data being vaulted? Does the DR Storage Class really contain **all** and **only** the VTVs to be used in DR? If you see DR VTVs that expire quickly after being sent offsite, you may be vaulting data that you don't need for DR. So you may want to analyze and manage the DR population to optimize MVC usage...just remember that this is a management effort, which costs *you* cycles.
8. Is the DR population mature? For short-term, quickly expiring DR VTVs, this consideration is irrelevant. However, at initial creation, long-term DR VTVs may be spread over multiple MVCs. After several cycles of the defragmentation process, however, they will be concentrated on a small number of offsite MVCs.
9. The `DELSCR` parameter setting also effects the number of MVCs used. For more information, see "DELSCR Considerations" on page 6.

Methods to Optimize DR MVC Usage

In "Vault Utilities MVC Usage Guidelines" on page 3, we gave you an overview of the factors that affect DR MVC usage, along with a few hints and tips to have it your way (safeguard your data and name that tune in *x* number of MVCs). Here's a summary of some ways to optimize MVC usage:

1. Send all immediate migrate DR VTVs to one (or at least fewer) VTSS(s). If that VTSS goes down, have an alternate set of `TAPEREQ`/`MGMTCLAS`/`STORCLAS` statements that send them to another VTSS.
2. **Do not** specify immediate migrate for DR VTVs. Instead, schedule a batch job that starts after all jobs creating DR VTVs have finished. The batch job migrates the DR VTVs by Management Class. Then run the vault utilities procedures, specifying in the parameters for `SWDIMS01` the Storage Classes to which you migrated the DR VTVs. **Note that** while this method improves MVC usage, it also **increases** the time between VTV creation and actual movement offsite.
3. **Do not** specify immediate migrate for DR VTVs. Instead, use the `CONSolid` utility to run VTV consolidation by Management Class after all VTVs have been created. Then run the vault utilities procedures, specifying in the parameters for `SWDIMS01` the Storage Classes specified in the `CONTGT` parameters for the various Management Classes. **Note that** while this method improves MVC usage, it also **increases** the time between VTV creation and actual movement offsite, and may cause extra I/O.

DELSCR Considerations

You use the DELSCR parameter of the MGMTclas statement to specify whether VSM deletes scratched VTVs, where DELSCR(YES) causes VSM to delete scratched VTVs, which frees VTSS buffer space and MVC space. Consider specifying DELSCR(YES) for the DR Management Classes. The following sections discuss the pros and cons of DELSCR(YES).

Pros

1. If you specify DELSCR(NO), the MVC does not become logically empty when the last VTV is scratched. Instead, the MVC *only* becomes logically empty when the last VTV on it is selected for rewrite, so that its logical location is no longer on this MVC. Because the Vault Utilities bring an MVC back onsite *only* when it is logically empty, the amount of time the MVC stays offsite is *increased*, and the total number of MVCs for the workload is *increased*.
2. For catalog controlled data sets on VTVs on MVCs--that is, MVCs with VTVs *without* programmatically determinable expiration dates--there is another consideration. The VTVs on these MVCs are duplexed, so that the VTV population of the offsite MVCs is mirrored by duplicate VTVs on MVCs which are located onsite. When the offsite MVCs become too fragmented, we drain them from the onsite copies. Note, however, that VTCS does *not* maintain duplicate copies of scratch VTVs, which comes into play when an onsite MVC is being reclaimed or drained. Normally, these two processes copy VTVs from MVCs and remigrate them. However, if the VTV has another, duplex copy, at remigration time VTCS just skips it, so that after the drain or reclaim, the only copy of the VTV is on the offsite MVC. Therefore, with DELSCR(NO) and catalog controlled data sets, when VTCS tries to drain the offsite MVC, it can't be mounted, and there is no onsite copy, so the drain process fails.

Cons

For VTVs with DELSCR(YES), there is a small window where data loss can occur in the scratch synchronization process as described in the following sections.

SLUCONDB Considerations - HSC

For HSC 5.1 and above, SLUCONDB has been enhanced to scratch *only* those volumes that are not in scratch status in the HSC CDS. Therefore, for HSC 5.1 and above, the *only* possibilities of inadvertently scratching a VTV resulting in data loss at scratch synchronization time are as follows:

- If you are running the HSC SLUADMIN Scratch Update Utility at the same time that SLUCONDB is running.
- If you do not specify the current TMS database and/or the current HSC CDS when using SLUCONDB.

For more information about HSC scratch synchronization with the Scratch Conversion Utility (SLUCONDB), see Chapter 5, “Utility Functions” of *HSC System Programmer’s Guide for MVS.*

Hint: These SLUCONDB enhancements are also available for previous releases of HSC as described in Table 6.

Table 6. HSC SLUCONDB Enhancements - PTFs for Previous Releases

HSC FMID	PTF Number
SOS4000	L1H119M
SOS4100	L1H119O
SOS5000	L1H119Q

Cons for ExLM SYNCVTV

1. For VTVs with DELSCR(YES), the only VTVs at risk are those that were already scratch in the HSC CDS when scratch synchronization began.
2. ExLM first makes a list of volumes to scratch by reading the HSC CDS.
3. ExLM next reads the TMS database records and, if SYNCVTV is in effect, compares the TMS scratch status and the HSC scratch status.
4. ***Only*** if the statuses are different, ExLM issues an HSC SCRATCH or UNSCRATCH operation to change the HSC scratch status so that it matches that of the TMC.
5. If a VTV has DELSCR(YES), HSC, when adding the volser to its scratch list, also deletes any existing instances.
6. For VTVs at risk (see Step 1., above), if they are selected for output between the time that list of volumes was made, and the time when their individual volsers are scratched, the data written on them is lost.

Hint: If you specify DELSCR(YES) for DR VTVs, ensure that jobs creating them **do not run** during the scratch synchronization window!

Installing the VSM Vault Utilities

Before you install the software:

- Ensure that you are at the latest service level for VTCS and its prerequisite software.
- Complete the pre-installation tasks described in the following sections:
  - “Verifying Installation Materials” on page 12
  - “Unloading the VSM Vault Utilities Install Jobs” on page 14
- If you have version 1.0 of the Vault Utilities installed in your NCS SMP/E zone and are planning on installing NCS 6.0 in the same NCS SMP/E zone as your existing NCS software, you must first delete version 1.0 of the Vault Utilities as described in “Deleting the 1.0 Version of the Vault Utilities” on page 10.

Next, install the VSM Vault Utilities as described in the following sections:

- “Allocating the CSI and General SMP/E Data Sets” on page 15
- “Setting Up the SMP/E Zones” on page 15
- “Allocating the Target and Distribution Libraries” on page 15
- “Receiving the VSM Vault Utilities FMIDs” on page 15
- “Apply Checking the VSM Vault Utilities FMIDs” on page 15
- “Applying the VSM Vault Utilities FMIDs” on page 15
- “Accepting the VSM Vault Utilities FMIDs” on page 16
- “APF Authorizing the VSM Vault Utilities Load Modules” on page 17

Deleting the 1.0 Version of the Vault Utilities

The following is required if you have Version 1.0 of the Vault Utilities installed in your NCS SMP/E zone and are planning on installing NCS 6.0 in the same NCS SMP/E zone. As described in the note on page 13 you **must** install 2.0 Vault Utilities FMIDs in a separate zone from NCS to avoid a conflict due to different SAS/C versions across the NCS and Offsite Vault products.

Use the SMP/E example JCL in Figure 1 on page 11 to delete the Version 1.0 of the Vault Utilities and its requisite functions. Modify the JCL to suit your environment; in particular a valid jobcard must be supplied and all values in lower case must be replaced. The JCL comments in the example give details on running the job.

```
//jobname  JOB  ....
//*
//*  THIS JOB DELETES THE VAULT UTILITY FUNCTIONS FROM THE NCS
//*  SMP/E ZONES.  ALL LOWER CASE VALUES MUST BE CHANGED TO
//*  VALUES SUITABLE FOR YOUR ENVIRONMENT.  THIS INCLUDES THE CSI
//*  NAME, THE NAMES OF THE NCS TARGET AND DISTRIBUTION ZONES AND
//*  THE REWORK LEVEL.
//*
//*  THE SMP/E COMMAND INPUT IS SET UP SUCH THAT THE JOB WILL
//*  NEED TO BE RUN SEVERAL TIMES, COMMENTING OUT AND
//*  UNCOMMENTING IN THE VARIOUS SMP/E COMMANDS AS NEEDED.
//*
//*  ALTERNATIVELY YOU CAN DELETE THE TWO CHECK COMMANDS AND
//*  UNCOMMENT THE REST OF THE COMMANDS AND RUN THE WHOLE PROCESS
//*  IN ONE RUN.
//*
//DELFUNC  EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=40M,
//         TIME=120,DYNAMNBR=120
//SMPCSI   DD   DSN=ncs.global.csi,DISP=SHR
//SMPOUT   DD   SYSOUT=*
//SMPRPT   DD   SYSOUT=*
//SMPLIST  DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SMPSNAP  DD   SYSOUT=*,FREE=CLOSE
//SMPWRK1  DD   UNIT=SYSDA,SPACE=(CYL,(100,100,600))
//SMPWRK2  DD   UNIT=SYSDA,SPACE=(CYL,(100,100,600))
//SMPWRK3  DD   UNIT=SYSDA,SPACE=(CYL,(100,100,600))
//SMPWRK4  DD   UNIT=SYSDA,SPACE=(CYL,(100,100,600))
//SMPWRK6  DD   UNIT=SYSDA,SPACE=(CYL,(100,100,600))
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(50,25))
//SYSUT2   DD   UNIT=SYSDA,SPACE=(CYL,(50,25))
//SYSUT3   DD   UNIT=SYSDA,SPACE=(CYL,(50,25))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(CYL,(50,25))
//SMPCNTL  DD   *
    SET BOUNDARY(GLOBAL).
      RECEIVE SYSMODS SELECT(FNCDELT)
 /* SET BOUNDARY(ncstgt).
     APPLY CHECK SELECT(FNCDELT) REDO */
 /* SET BOUNDARY(ncstgt).
     APPLY SELECT(FNCDELT) REDO */
 /* SET BOUNDARY(ncsdlib).
     ACCEPT CHECK SELECT(FNCDELT) REDO */
 /* SET BOUNDARY(ncsdlib).
     ACCEPT SELECT(FNCDELT) REDO */
    .
/*
//SMPPTFIN DD   *
++ FUNCTION(FNCDELT) REWORK(200nnnn).
++ VER(Z038) DELETE(
                  SWD4000  /* VAULT UTILITY */
                  SSKQ230  /* COMMON PARSER */
                  ASAR55D  /* SAS/C RUNTIME */
                  ASAT55D
                  ).
/*
```

Figure 1. ***Example JCL to Delete Vault Utilities Version 1.0***

Verifying Installation Materials

Before installing the VSM Vault Utilities, make sure you have the VSM Vault Utilities product tape.

The VSM Vault Utilities Installation Tape Contents

The VSM Vault Utilities is delivered on the VSM Vault Utilities product tape that contains the VSM Vault Utilities FMIDs. Table 7 lists the files included on the VSM Vault Utilities product tape.

Table 7. The VSM Vault Utilities Product Tape Contents

File	Data Set Name	Description
1	SMPMCS	SMP/E control statements
2	SWD6000.F1	SWD6000 JCLIN and SMP/E install sample JCL jobs: **VDRCSI** - Sample job to allocate the CSI and general SMP/E data sets. **VDRZONES** - Sample job to define the SMP/E zone structure. **VDRDDEF** - Sample job to allocate target and distribution libraries. **VDRRECV** - Sample job to receive the VSM Vault Utilities FMID. **VDRAPPLY** - Sample job to apply check/apply the VSM Vault Utilities FMID. **VDRACCPT** - Sample job to accept the VSM Vault Utilities FMID.
3	SWD6000.F2	SWD6000 object modules
4	SWD6000.F3	SWD6000 MACLIB and SAMPLIB members (SAMPLIB members are automatically installed in the NCS SAMPLIB); for more information, see “VSM Vault Utilities SAMPLIB Members” on page 17.
5	SSKQ230.F1	SSKQ230 JCLIN
6	SSKQ230.F2	SSKQ230 object modules
7	ASAR55D.F1	ASAR55D JCLIN
8	ASAR55D.F2	ASAR55D object modules

The VSM Vault Utilities FMIDs

The VSM Vault Utilities software is packaged in standard SMP/E format. The VSM Vault Utilities installation tape includes the following FMIDs:

`SWD6000`
: contains the VSM Vault Utilities load modules.

`SSKQ230`
: contains the parser load modules.

`ASAR55D`
: contains the SAS/C runtime libraries.

Note: You must install these FMIDs in a separate zone from NCS to avoid a conflict due to different SAS/C versions across the NCS and Offsite Vault products.

Unloading the VSM Vault Utilities Install Jobs

SWD6000.F1 contains sample install jobs for the VSM Vault Utilities. Figure 2 shows JCL to unload these install jobs.

```
//            ********************************************************************
//            *            JOB:          jobname
//            *
//            *            PURPOSE:      LOAD THE SMP/E INSTALL SAMPLE JCL
//            *
//            *            CHANGES :     COMPLETE THE JOB STATEMENT,
//            *                          FILL IN THE DSN FOR SYSUT2.
//            *
//*           ********************************************************************
//jobname     JOB ......
//JCL         EXEC          PGM=IEBCOPY
//SYSPRINT    DD            SYSOUT=*
//SYSUT1      DD            DSN=SWD6000.F1,DISP=OLD,
//            UNIT=tapeunit,VOL=SER=WD6000,LABEL=(2,SL)
//SYSUT2      DD            DSN=your.dataset.name,
//                          DISP=(,CATLG,DELETE),
//                          UNIT=SYSALLDA,
//                          SPACE=(TRK,(5,1,5))
//SYSIN       DD            *
 C INDD=SYSUT1,OUTDD=SYSUT2
 E M=SWD6000
//*
```

Figure 2. JCL to Unload the VSM Vault Utilities Installation Jobs

Allocating the CSI and General SMP/E Data Sets

Run SAMPLIB member VDRCSI to allocate the CSI and general SMP/E data sets (SMPPTS, and so forth).

Setting Up the SMP/E Zones

Run SAMPLIB member VDRZONES to set up the SMP/ E zone structure and define the DDDEFs for the general data sets allocated by VDRCSI.

Allocating the Target and Distribution Libraries

Run SAMPLIB member VDRDDEF to allocate the target and distribution libraries and run the UCLIN for the target and distribution library zones.

Receiving the VSM Vault Utilities FMIDs

Run SAMPLIB member VDRRECV to receive the VSM Vault Utilities FMID.

Apply Checking the VSM Vault Utilities FMIDs

Run SAMPLIB member VDRAPPLY to apply check the VSM Vault Utilities FMID.

Applying the VSM Vault Utilities FMIDs

Delete the CHECK statement from SAMPLIB member VDRAPPLY, then run this member again to apply the VSM Vault Utilities FMID. When the apply is successful, the SMP/E target libraries contain the data sets described in Table 8.

Table 8. SMP/E Target Library Contents

Data Set Name	Contents
yourhlq.SWD6000.SWDLINK	Load modules required for VSM Vault Utilities execution
yourhlq.SWD6000.SWDSAMP	Sample material for use with VSM Vault Utilities.

Note: There are other LOADLIBs allocated in the VDRDDEF job and used by the APPLY process. When run without the CHECK option, the VDRAPPLY job generates a return code 4, even though the job is successful.

Accepting the VSM Vault Utilities FMIDs

Run SAMPLIB member `VDRACCPT` to accept the VSM Vault Utilities FMID. When the accept is successful, the SMP/E distribution libraries contain the data sets described in Table 9.

Table 9. SMP/E Distribution Library Contents

Data Set Name	Contents
`yourhlq.SWD6000.ASWDLINK`	Load modules required for VSM Vault Utilities execution
`yourhlq.SWD6000.ASWDSAMP`	Sample material for use with VSM Vault Utilities.

APF Authorizing the VSM Vault Utilities Load Modules

You must APF authorize the VSM Vault Utilities Load Modules (`yourhlq.SWD6000.SWDLINK`).

VSM Vault Utilities SAMPLIB Members

The VSM Vault Utilities installation automatically installs in the Vault Utilities SAMPLIB the members described in Table 10.

Table 10. VSM Vault Utilities SAMPLIB Members

EXL1SAMP	ExLM sample job to key on vault code assigned to MVCs and eject MVCs with message for vaulting offsite.
EXL2SAMP	ExLM sample job to produce a report file that selects MVCs that are empty. This file is input to `SWDIMS02`, which processes the MVCs for return from offsite.
EXL3SAMP	ExLM sample job to produce a report file that selects MVCs that are less than a specified percent used. This file is input to `SWDIMS02`, which processes the MVCs for defragmentation.
SWD1SAMP	`SWDIMS01` sample job to vault MVCs in the specified Storage Class.
SWD2SAMP	`SWDIMS02` sample job to return MVCs from offsite as selected by `EXL2SAMP`. A second function is to facilitate defragmenting MVCs as selected by `EXL3SAMP`.
PRMSAMP1	Sample parameter file that specifies the Storage Classes and other processing options for the `SWDIMS01` utility.
PRMSAMP2	Sample parameter file that specifies the days until the MVC expires for the `SWDIMS02` utility.

Offsite Vaulting - The Pickup Truck Access Method for CA-1

This section tells how to use VSM and the PTAM ("Pickup Truck Access Method") to disaster-proof a single-site, offsite vault data processing operation so it can continue and resume operations after a catastrophic failure to the system. Throughout the procedures, you'll see how to do this for data sets with retention periods and for those without retention periods...specifically, they are catalog controlled data sets. The additional sub-flavor is that we're using a TMS to extract the date stamps, and the TMS is CA-1. The advantage of using a TMS is that there are accompanying StorageTek-written utilities that help automate the whole process.

As Figure 3 on page 21 shows, our example configuration consists of an MVS host at the Production Site, attached VTSS, single ACS, and an Offsite Vault. The vaulting of critical data works as follows:

- Data sets **with** retention periods are routed via `TAPEREQs` to VTVs by a series of Management Classes (keyed to retention periods) that specify `MIGRATE IMMEDIATE (DELETE)`. Each Management Class specifies a corresponding Storage Class so that the VTVs for that retention period are grouped on a common set of MVCs. After the migrations complete, you next run the `SWDIMS01` utility to create export MVCs to eject via ExLM for transport to the offsite vault. The VTVs on the export MVCs have expiry dates per the TMS (CA-1). As these VTVs expire, you use an ExLM Custom Volume Report and the `SWDIMS02` utility to identify MVCs with no current VTVs in the offsite vault. You then use CA-1 VMS batch jobs to create a list of these MVCs to return them to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs with a current data generation.
- **Catalog controlled** critical data sets are routed via a `TAPEREQ` to VTVs by a single Management Class that specifies `MIGRATE IMMEDIATE (DELETE)`. This Management Class specifies a two corresponding Storage Classes, one for onsite storage, one for offsite vaulting. After the migrations complete, you next run the `SWDIMS01` utility against the offsite Storage Class to create export MVCs to eject via ExLM for transport to the offsite vault. When ExLM reports MVCs with space utilization of less than a user-specified value, you use the `SWDIMS02` utility to identify these MVCs. You then "logically drain" these MVCs (by draining the onsite MVC), use CA-1 VMS batch jobs to create a pull list of the MVCs eligible for reuse, and then return the offsite MVCs to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs.

Caution: Please note the following gotcha. Some customers use HSC User Exit 06 to enter new tapes into the ACS and define them as scratch. This is exactly what we'd want for Nearline tapes, but scratch MVCs are a **not** a value add. So if you're entering MVCs and HSC User Exit 06 is set to define newly entered tapes as scratch, use the HSC `UEXIT` command to disable the exit before entering tapes to be used as MVCs.

"Offsite Vaulting - The Pickup Truck Access Method for CA-1" consists of the following subsections:

- "Normal Operations" on page 21, where we show what the Production Site looks like when the data is flowing smoothly. Here, we also tell how to set up the system to withstand a failure.
- "Business Continuance" on page 38, which is how to stay open for business if the Production Site takes a major hit.
- "Business Resumption" on page 41, which is how to get back to normal operations once the Production Site is up and running again.

Normal Operations uses the StorageTek VSM Vault Utilities (the `SWDIMS01` and `SWDIMS02` utilities), so before diving into Normal Operations, let's turn the page and install these utilities.

Normal Operations

On page 19, we gave an overview of the system, disaster recovery needs, and operations to meet those needs. Figure 3 and Figure 4 on page 22 show the system during normal operations for VTV migration and recall. Figure 5 on page 22 shows the export and physical transport to the offsite vault of the MVCs that contain the critical data. There is, as we said, Some Assembly Required, so let's turn to "Migration, Export, Offsite Vault, and MVC Reuse" on page 23 for the details.

Normal Operations - VTV Migration

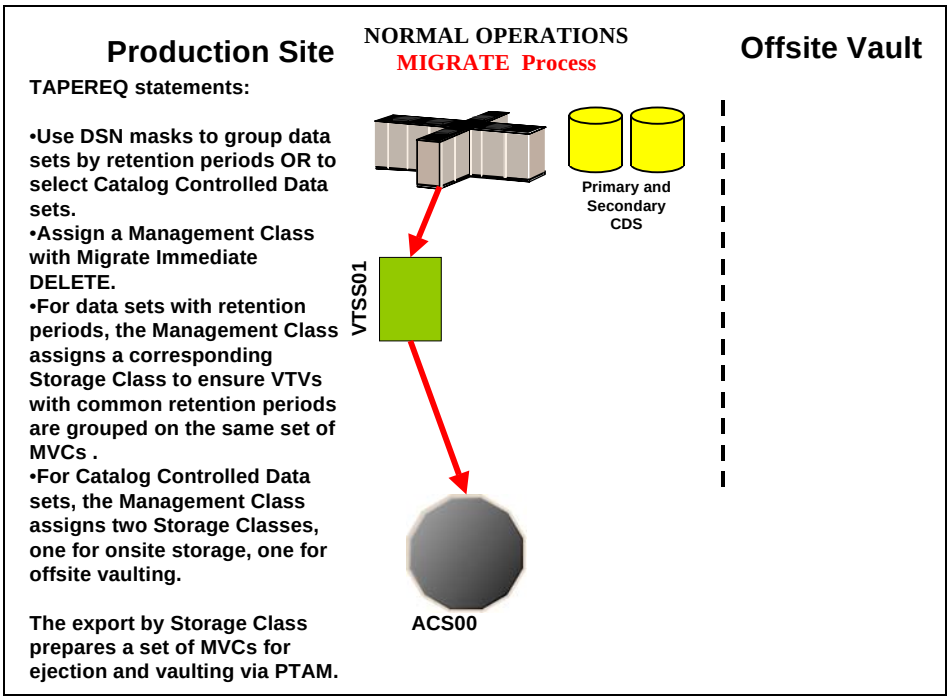


Figure 3. Normal Operations - VTV Migration

Normal Operations - VTV Recall

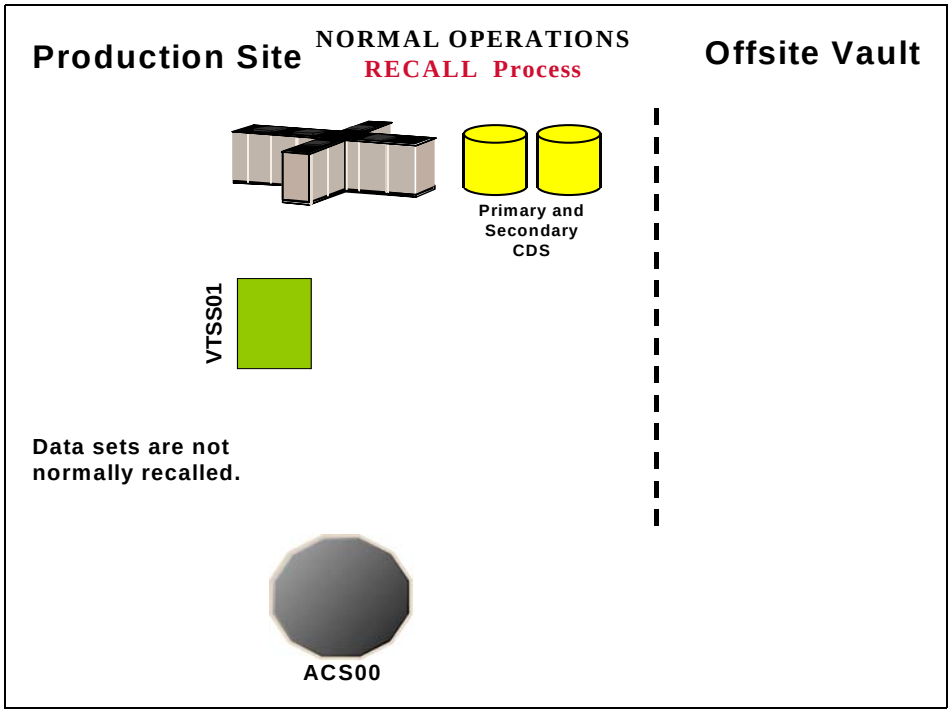


Figure 4. Normal Operations - VTV Recall

Normal Operations - MVC Export

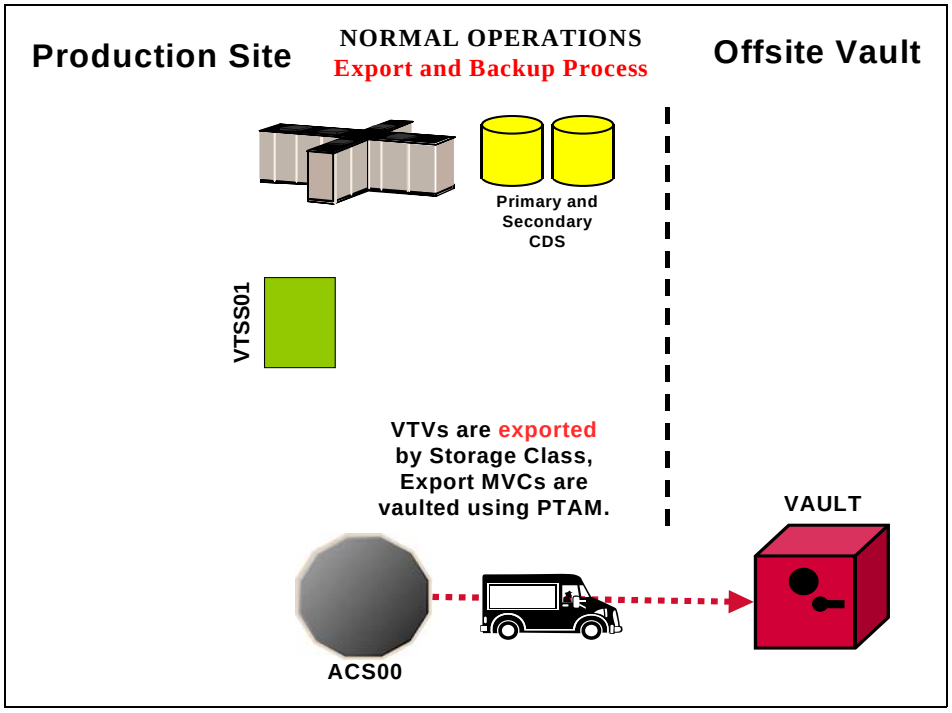


Figure 5. Normal Operations - MVC Export

Migration, Export, Offsite Vault, and MVC Reuse

In “Normal Operations” on page 21, we showed the system under normal operations. This section tells how to route critical data **in data sets with retention periods** and **catalog controlled data sets** to VTVs, migrate them to a common set of MVCs, and vault the MVCs offsite.

Table 12 on page 24 and Table 11 on page 24 are about two distinctly different kinds of data sets:

- First, in Table 11 on page 24, for **catalog controlled data sets**, we’re again showing the use of a data set name mask on TAPEREQ. As above, use other data set characteristics that you specify on TAPEREQ if that works better. Note that with catalog controlled data sets, the TAPEREQ specifies a single Management Class that duplexes the data via two Storage Classes. This produces an offsite MVC and an onsite MVC with the critical data, which, as we’ll see later, is critical to reusing and recycling MVCs.
- In Table 12 on page 24, for **data sets with retention periods**, we’re showing a situation where you have multiple data set name masks, each of which corresponds to a different retention period for the data sets selected by that mask. The fewer of these, the better (read what the VTCS documentation says about Too Many Storage Classes). That is, for *each* data set name mask you need a corresponding Management Class and Storage Class.

  With TAPEREQ, in fact, you have lots of choices...for example, you can key on jobname or stepname, if that works better. If you want to key on retention periods, data set name is probably the right move, however. We just want to make sure you know that you **do** have the flexibility to key on different data set groupings with different characteristics.

So at this point your question is “What do I do if I hav***e both*** flavors of data sets?” Well, we thought of that, too, because in the real world this is a very likely situation. Back in the MVC reuse section, which is one of the key areas to get right, we break that part of the process down for you. You can read about all of that in “Returning MVCs from the Offsite Vault for Reuse” on page 31. Another useful tool, however, is the overall process flow for “mixed” shops that we show in “A Sample Offsite Vaulting Work Flow” on page 173.

- In this example, we're selecting the MVCs for offsite vaulting from a Named MVC Pool. This is optional, but it's not a bad idea because:
  - All the non-critical jobs select MVCs either from another Named MVC Pool or from the overall MVC Pool (except for the Named Pool used for offsite vaulting).
  - At the Recovery Site, you only have to define the MVCs in the Named Pool.

  Once again, however, read the *caveats* in the VTCS documentation about the potential gotchas with Named MVC Pools.
- Finally, throughout the procedures, we refer you back to Table 12 or Table 11. In fact, what we *really* mean is your versions of these tables, which you'll have to create to match your shop's situation...as recorded in "Offsite Vaulting Configuration Record" on page 177.

Table 11. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Catalog Controlled Data Sets without Retention Periods

Management Class	**Storage Classes**	**Use**	**Named MVC Pool**	**TAPEREQ Data Set Mask**
mgmtcat	storon storoff	On site MVCs Offsite MVCs	poolcat	maskcat

Table 12. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Data Sets with Retention Periods

Management Class (mgmt-class-name)	**Retention Period RETPD(LT,n)**	**Storage Class (stor-clas-name)**	**Named MVC Pool (poolname)**	**TAPEREQ Data Set Mask (mask)**
mgmtret1	1 to 7 days	storret1	*poolret1*	maskret1
		. . .		
mgmtretn	Greater than one century	storretn	*poolretn*	maskretn

- Let's not forget the SWDTMS01 and SWDTMS02 utilities, which we'll use in the following sections. For some reference information, see "SWDTMS01 Utility" on page 144 and "SWDTMS02 Utility" on page 155.

  The whole of Normal Operations takes some doing, so let's break it down into the following sections:

  - "Setting Up the System" on page 25
  - "Creating the Export MVCs and Vaulting Them Offsite" on page 29
  - "Returning MVCs from the Offsite Vault for Reuse" on page 31

Setting Up the System

To set up the system:

1. **First, we'll enable the Advanced Management Feature.**

   The Advanced Management Feature is required to implement Storage Classes, run the EXPORT and IMPORT utilities, and use selected Management Class parameters (such as MIGPOL).

2. **Let's build the Management Classes that assign a corresponding Storage Class and do an immediate migrate with delete.**

   - **For data sets with retention periods**, *each* of the MGMTclas statements (you create a separate statement for each retention period) looks something like Figure 6:

```
MGMT NAME(mgmtretn) IMMED(DELETE) MIGPOL(storretn) DELSCR(YES)
```

Figure 6. Management Classes for VTVs with Critical Data

   - **For catalog controlled data sets**, the MGMTclas statement looks something like Figure 7:

```
MGMT NAME(mgmtcat) IMMED(DELETE) MIGPOL(storon,storoff) DELSCR(YES)
```

Figure 7. Management Classes for VTVs with Critical Data - Catalog Controlled Data Sets

   In Figure 6 and Figure 7, each Management Class specifies immediate migrate delete, which is designed to immediately put VTVs in this Management Class on the migration queue and delete the VTSS copy once it is migrated. This ensures quick migration and frees the VTSS buffer. We also assign the corresponding Storage Class to the MVCs that contain the migrated VTVs as shown in Table 12. on page 24 or Table 11 on page 24.

Warning: Note that for both types of data sets, the Management Class specifies DELSCR(YES). If you haven't already done so, **go back and read** the planning information in "DELSCR Considerations" on page 6.

3. **Next, we'll create the Storage Classes that own the MVCs that contain the migrated VTVs.**

   - **For data sets with retention periods**, each of the STORclas statements (you create a separate statement for each retention period) looks something like Figure 8:

```
STOR NAME(storretn) MVCP(poolret)
```

Figure 8. Storage Classes for Data Sets with Retention Periods

   For the Storage Class names to plug in, see Table 12. on page 24. In Figure 8, the STORclas statement associates each Storage Class with the specified Named MVC Pool.

   - **For catalog controlled data sets**, the STORclas statements looks something like Figure 9:

```
STOR NAME(storon) MVCP(poolcat)
STOR NAME(storoff) MVCP(poolcat)
```

Figure 9. Storage Classes for Catalog Controlled Data Sets

   For the Storage Class names to plug in, see Table 11. on page 24. In Figure 9, the STORclas statements associate each Storage Class with the specified Named MVC Pool.

4. **Next, we'll use the MGMTDEF command to load the MGMTclas and STORclas control statements we created in Step 2 on page 25 and Step 3 on page page 26.**

5. **Now let's create TAPEREQ statements to route the critical data to VSM and assign the corresponding Management Class to the data.**

   - **For data sets with retention periods**, each of the TAPEREQ statements (you create a separate statement for each retention period) looks something like Figure 10:

```
TAPEREQ DSN(maskretn) MEDIA(VIRTUAL) MGMT(mgmtretn) RETPD(LT,n)
```

Figure 10. TAPEREQ Statements for Data Sets with Retention Periods

   Note: The RETPD parameter is not supported under JES3, so use another TAPEREQ data set selection method (for example, JOBNAME).

   See Table 12. on page 24 for the correct values to fill in for Figure 10.

   - **For catalog controlled data sets**, the TAPEREQ statement looks something like Figure 11:

```
TAPEREQ DSN(maskcat) MEDIA(VIRTUAL) MGMT(mgmtcat)
```

Figure 11. TAPEREQ Statement for Catalog Controlled Data Sets

   See Table 11. on page 24 for the correct values to fill in for Figure 11.

6. **Use the TREQDEF command to load the TAPEREQ control statements we created in Step 5.**

7. **If they aren't already in place, set up the CA-1 vault codes for the exported MVCs.**

   CA-1 uses the data set name (and optionally the name of the job that creates the data set) to assign vaulting codes. For more information on setting up vaulting codes to achieve the desired vaulting assignments and rotation, see *CA-1 User's Guide*.

   The name of the data set that the `SWDIMS01` utility assigns to the MVCs in the TMC is:

   ```
   DSN=prefix.storclas.MVCTAPE.RESERVED
   ```

   Where:

   - `prefix` is the HLQ specified in the `MvcDsnPreFiX` parameter of the `STORclas` statement. For more information, see "STORCLAS" on page 169.
   - `storclas` is a Storage Class created in Step 3 on page 26.

   For example, if you specified a prefix of `SYS1.VSMDR` on the `MvcDsnPreFiX` parameter and `LE8DAYS` for the Storage Class for retention periods of less than 8 days, the `SWDIMS01` utility assigns to the MVCs for that Storage Class the following DSN:

   ```
   SYS1.VSMDR.LE8DAYS.MVCTAPE.RESERVED
   ```

   Similarly, if you specified a prefix of `SYS1.VSMDR` on the `MvcDsnPreFiX` parameter and `CATOFF` for the Storage Class for "offsite" MVCs with catalog controlled data sets, the `SWDIMS01` utility assigns to the MVCs for that Storage Class the following DSN:

   ```
   SYS1.VSMDR.CATOFF.MVCTAPE.RESERVED
   ```

8. **There is one last item...**

   ...which is defining the MVCs to the TMS. As it says in the VTCS documentation:

   > "Access to the MVCs via an RTD bypasses the MVS intercepts put in place by the tape management system so that it does *not* record within its database any access to the MVCs by VSM and does *not* automatically provide protection against inadvertent overwrites of non-expired data on MVCs. Therefore, if you choose to define MVCs to the tape management system, StorageTek **strongly recommends** that you define them as non-scratch, non-expiring volumes."

   Well, that isn't going to work so well for MVCs that you want to reuse, so we have to find another way. So if you want to define the MVCs to CA-1 and adequately protect them, try this:

   a. Do a format extend to add the MVC volsers to the TMC.

   b. Second, define the MVCs as a scratch subpool to CA-1.

   c. Finally, write a rule that says that *only* HSC/VTCS can write to the MVCs as scratches.

   Section 1, Setting Up the System, is done, so let's proceed to "Creating the Export MVCs and Vaulting Them Offsite" on page 29.

Creating the Export MVCs and Vaulting Them Offsite

To create the export MVCs and vault them offsite:

1. **Ensure that all VTVs with critical data are migrated.**

   The TAPEREQ statements in Step 5 on page 26 are the triggers to start the VTV migrations to the MVCs we'll export. But how do we know when all the VTVs with critical data are migrated? Basically, we write a "watchdog" program that monitors VTV migration and reports when the *last* critical VTV is migrated.

2. **Run the SWDIMS01 utility.**

   The SWDIMS01 utility, working together with the JCL you create to invoke it, does the following:

   - **Runs the EXPORT utility** against **all** Storage Classes we created in Step 3 on page 26 **and creates a comprehensive Manifest File** that includes all these Storage Classes. We use the STORclas statement of the parameter file to specify one or more Storage Classes. For more information, see "STORCLAS" on page 169.

   Note: If an MVC in a specified Storage Class is in use when the SWDIMS01 utility runs, it will be skipped and will not be exported until the next time the utility runs.

   The comprehensive Manifest File is used, if needed, in Business Continuance mode (see Step 4 on page 38)...so it's probably a good idea to have one copy of the file on disk and another on tape stored offsite.

   - **Creates an MVC update file**, which is used as input to TMSUPDTE in Step 3 on page 30, to set data set names and expiration dates for the MVCs to be sent offsite. These commands set the data set expiration date to PERMANENT.
   - **Creates a VTV update file**, which can optionally be used as input to TMSUPDTE in Step 3 on page 30, to change the TMS VTV records.

     These control cards reflect the state of the VTVs being sent offsite, and it's probably not a good idea to alter them. If, for some reason, you have a need to change the state of VTVs being sent offsite, contact StorageTek Software Support for information on the use of these cards.

Hint: Good Practices suggests that we **only** want to vault volumes that are resident in an ACS. To do this, run the SWDTMS01 utility with the STORCLAS ALL(OFF) parameter of the parameter file. If, however, you want to see a selection list of all volumes that **should** be vaulted, whether they are ACS resident, use the STORCLAS ALL(ON) parameter. For more information, see "STORCLAS" on page 169. **Note that**, even if you specify the ALL(OFF) parameter, the Manifest File is **always** "comprehensive."

For example JCL for the SWDTMS01 utility, see "SWDTMS01 JCL Example" on page 146.

3. **Next, we run TMSUPDTE to update the TMC with MVC information.**

   TMSUPDTE uses the input from Step 2 on page 29 to update the MVC volume records in the TMC with a dummy data set name, expiration date, creation date, and so forth.

Hint: In Step 2, if the SWDTMS01 utility **does not** select any MVCs for update, SWDTMS01 ends with a return code of 4. Because the TMC update in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDTMS01 runs successfully, selects MVCs, and completes with a return code of 0 as shown in "SWDTMS01 JCL Example" on page 146.

4. **To assign the vault codes and rotation to the MVCs, run the CA-1 VMS batch jobs.**

   ...where we set up the vault codes in Step 7 on page 27 and updated the TMC with MVC information in Step 3 on page 30.

   For more information on the CA-1 VMS, see *Brightstor CA-1 Tape Management Utilities and Reports Reference*.

5. **Keying on the Vault Code from Step 7 on page 27, we use ExLM to eject the MVCs from the ACS.**

   Figure 81. on page 149 shows example JCL to use ExLM to eject the MVCs.

6. **Finally, we use PTAM (Pickup Truck Access Method) to do the offsite vaulting of the MVCs we ejected in Step 5.**

   Because we have a finite number of MVCs, when VTVs go non-current on exported MVCs, we need a way to cycle these MVCs back into the system for reuse. For that procedure, see "Returning MVCs from the Offsite Vault for Reuse" on page 31.

Returning MVCs from the Offsite Vault for Reuse

In “Creating the Export MVCs and Vaulting Them Offsite” on page 29, we migrated VTVs with critical data to MVCs and exported those MVCs to the Offsite Vault. We don’t have an infinite number of MVCs for vaulting, so we need a way to recycle vaulted MVCs back into the system when some or all VTVs on these MVCs go non-current. And there are three flavors, so pick the one that works for your shop and follow the referenced procedure:

1. **Flavor 1 - Sites with *only* data sets with retention periods**, when ExLM reports that all VTVs on an MVC have reached their expiry dates in CA-1, the current VTV count goes to zero and we can return the MVC to the Production Site for reuse. For procedures, see “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 35.

2. **Flavor 2 - For sites with *only* catalog controlled data sets**, we use an ExLM report to look for vaulted MVCs that have a usage percentage less than a percentage we specify. In other words, ExLM looks for MVCs that have enough free space to make it worthwhile to reuse these MVCs. What’s a reasonable percentage to specify? The answer, of course, is “it depends on the needs of your shop.” Specify a high percentage and you’ll free up lots of MVCs...at the cost of high VSM/system activity. Specify a low percentage and you’ll have lots of fragmented MVCs in the offsite vault. Try starting with 25% and adjust as needed.

   The way we reuse these MVCs is...well, it’s complex, so let’s slow down and see how it works:

   1. MVCs with less than `n` percent usage are marked `LOST`. Now why would I do that? Because I have an equivalent on site copy and I don’t want to put an operator through mounting the offsite MVCs to drain it, see number 2.

   2. I try to run a drain against the offsite MVC, it’s lost, so instead VTCS “logically drains” the offsite MVC by recalling all its VTVs from the MVC copy that is onsite. Because the recalled VTVs have the DR Management Class, they are migrated to a new MVC that can be taken offsite. The new MVC is now the input to Step 2 on page 29, which detects MVCs with Storage Classes that require offsite vaulting.

      So VTCS “logically drained” the offsite MVC, which is goodness because the offsite MVC is now 0% full. It is now eligible for selection by Step 1 on page 35 and is returned per Step 7 on page 36.

   And for the step-by-step implementation of this process, see “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 33.

Note that you are not done until, as it says in Step 4 on page 34, you next complete the procedure described in “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 35...because the logical draining has created MVCs with a zero current VTV count. Is that cool, or what?

Hint: How often do you want to do this defragmentation? The answer is “it depends on the needs of your shop and your level of vaulting/reuse activity”...but doing this once a week is probably a good starting point.

3. **Flavor 3 - For "mixed" sites with *both* catalog controlled data sets *and* data sets with retention periods,** you do the same thing (and for the same reasons) that you do for sites with only catalog controlled data sets. That is, you first do the procedure in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 33, followed by the procedure in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 35.

Note: Each of the three procedures described above **requires** the standard VMS update process described in *Brightstor CA-1 Tape Management Utilities and Reports Reference*.

Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets.

To Recycle Offsite MVCs That Contain VTVs With Catalog Controlled Data Sets:

1. **First, let's run an ExLM Custom Volume Report...**

   ...to find MVCs with less than a specified percent used.

   Figure 82. on page 157 shows a JCL example that generates an ExLM report that selects vaulted MVCs with less than a specified percent used:

   - `DSN=prefix.storoff.MVCTAPE.RESERVED` is the name of the data set that the `SWDTMS01` utility assigns to the MVC in the TMC, as described in Step 7 on page 27.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "STORCLAS" on page 169.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specify in Step 1 on page 35. The difference is in the *storclas* value for the middle qualifier on the DSN. In this step, it is the single DSN for the data set assigned to all MVCs that contain VTVs with catalog controlled data sets. In this step, you want to drain ***only*** the MVCs with catalog controlled data sets, not the MVCs with data sets with retention periods. If you go all the way back to Step 3 on page 26, you'll see that it is, in fact, `storoff`, which is the Storage Class for the offsite (vaulted) MVCs.

   - `loccode` is the location code for the offsite vault.
   - `percent` is the specified percentage use. See page 31 for a discussion of what constitutes a reasonable value.
   - `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDTMS02` utility to process the flat file report generated in Step 1.**

   "JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with Less Than a Specified Percent Use" on page 159 shows example JCL where the `SWDTMS02` utility:

   - Creates the appropriate `MVCMAINT` commands to mark as `LOST` the MVCs selected in Step 1.
   - Creates the appropriate `MVCDRAIN` commands to make all the swell stuff happen that we described starting in number 1. on page 31.

3. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 33 to enable draining the fragmented offsite MVCs.**

Caution: Here we need to take a look at "JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with Less Than a Specified Percent Use" on page 159. This is the JCL to run the SWDTMS02 utility in Step 2 on page 33. Note that the JCL specifies a DD that contains "simplexed" output cards (SWDSIMP) and a DD that contains "duplexed" output cards (SWDDUP). In this step, you **must** run MVCMAINT against the duplexed cards (SWDDUP)!

Also note that, in Step 2, if the SWDTMS02 utility does not select any MVCs for update, SWDTMS02 ends with a return code of 4. Because the SWSADMIN commands in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDTMS02 runs successfully, selects MVCs, and completes with a return code of 0.

"JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with Less Than a Specified Percent Use" on page 159 shows example JCL to run SWSADMIN against the duplexed cards only if MVCs are selected.

Finally, **Also note that**...we're running a drain to recall the current VTVs from the MVC...remember that these are MVCs that are x percent used. How does this actually happen? For the long version, see page 31. Here's another data point:

> After Step 3 completes, the still current VTV, which had one copy on the offsite MVC which was drained, will have one copy on the onsite MVC, and a new copy on another onsite MVC which has the same Storage Class as the original offsite MVC. In the meantime, the offsite MVC will have had its storage class removed. This new onsite DR MVC will be picked up in the next SWDTMS01 run, and will be sent offsite.

And, additionally:

> The drain process also clears the LOST status of the offsite MVC.

4. **Don't stop now...**

...because, for all the good reasons we gave you back on page 31, this process **feeds into** and **requires** that you next complete the procedure described in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 35.

Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods.

To Recycle Offsite MVCs That Contain VTVs With Data Sets with Retention Periods:

1. **First, let's run an ExLM Custom Volume Report to find any MVCs vaulted for data sets with retention periods with no current VTVs....**

   "JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with No Current VTVs" on page 161 shows a JCL example that generates an ExLM report that selects vaulted MVCs with no current VTVs:

   - `DSN=prefix.storclas.MVCTAPE.RESERVED` is the name of the data set that the `SWDTMS01` utility assigns to the MVC in the TMC, as described in Step 7 on page 27.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "STORCLAS" on page 169.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specified in Step 1 on page 33. The difference is in the *storclas* value for the Second Level Qualifier on the DSN. In this step, we want to detect all offsite MVCs with 0 VTVs. Therefore, the DSN mask in the ExLM job should be "`prefix.**`".

   - `loccode` is the location code for the offsite vault.

   `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDTMS02` utility to process the flat file report generated in Step 1.**

   "JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with No Current VTVs" on page 161 shows example JCL where the `SWDTMS02` utility:

   - Creates the TMS `TMSUPDTE` commands to modify TMS volume records for MVCs. These commands set the Expiration Date specified on the `EXPIRYPERIOD` statement to allow a return of the MVC from the vault. For more information, see "EXPIRYPERIOD" on page 167.
   - Creates the appropriate `MVCMAINT` commands to reset the Readonly bit to make these MVCs available for reuse after they're reentered into the ACS.
   - Creates the appropriate `MVCDRAIN` commands to...trust us, it's a good idea, and we'll explain it all in Step 4 on page 36.

3. **Run the CA1 TMSUPDTE program to process the control cards generated in Step 2...**

   ...to update the TMC accordingly. We do this, together with some other fun SWSADMIN stuff, as shown in the example in “JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with No Current VTVs” on page 161.

4. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 35 to reset the readonly status of the MVCs that will be returning to the production site...**

   ...and to also do a logical drain on those MVCs. “Now why,” you ask yourself, “Is he telling me to drain an *empty* MVC?” And the answer is that invoking MVCDRAIN does more than just draining VTVs, it also affects an MVC state change, which in this case is a Good Thing. Specifically, the MVDRAIN operation removes the Storage Class from the MVC, which prevents the MVC from being selected when SWDIMS01 is exporting MVCs...and also makes the MVC available to the universe, not just that Storage Class. Don’t worry, these empty MVCs will not have to be mounted to be logically drained.

Caution: Here we need to take a look at Figure 86 on page 161. This is the JCL to run the SWDIMS02 utility in Step 2 on page 35. Note that the JCL specifies a DD that contains “simplexed” output cards (SWDSIMP) and a DD that contains “duplexed” output cards (SWDDUP). In this step, you **must** run MVCMAINT against the simplexed cards (SWDSIMP)!

Also note that, in Step 2 on page 35, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in this step will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

Figure 87 on page 162 shows example JCL to run SWSADMIN against the simplexed cards only if MVCs are selected.

5. **Run the CA1 VMS batch job(s) to create the list of MVCs to be returned from offsite storage for reuse.**

6. **Using VMS picklists, do the PTAM thing to take the MVCs ejected in Step 5 on page 30 to the offsite vault.**

7. **While you’re at the offsite vault...**

   ...pick up any MVCs which are to be moved back to the onsite library. This is a little hard to visualize, so think of it this way: We actually identified these MVCs in Step 2 on page 35 during **a previous run** of this procedure that occurred **greater than** nn **days ago**, where nn is the expiry period specified on the EXPIRYPERIOD statement. For more information, see “EXPIRYPERIOD” on page 167.

Great, we're done...that is, until the next time we get to cycle back into the defrag procedure described in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 33.

Business Continuance

Okay, the Unthinkable has occurred, and the Production Site has taken a major hit. Now what? Well, the operations staff next wants to switch to business continuance mode. Note that because of the setup we did in “Normal Operations” on page 21, we can quickly and effectively resume operations because:

- All critical data is migrated and vaulted.
- A Recovery Site (which could be a vendor such as Comdisco) is standing by with VSM installed.

Because we did our homework on the setup, the switch to Business Continuance Mode at the Recovery Site is quick and straightforward via the MVC import shown in Figure 12 on page 39. After we switch to Business Continuance Mode, operations look like Figure 13 on page 39 and Figure 14 on page 40. We’ll use the following procedure to switch to Business Continuance Mode.

To switch to Business Continuance Mode at the Recovery Site, do the following:

1. **Create a new CDS at the Recovery Site.**

   This CDS reflects LIBGEN and VTCS configuration at the Recovery Site; we’ll populate this with VTV and MVC information in Step 4.

2. **Enter only critical MVCs into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

3. **Start an HSC audit on the panels we filled in Step 2.**

4. **Using the *most current* comprehensive Manifest File, run an import into the *new* CDS we created in Step 1.**

   We created the comprehensive Manifest File in Step 2 on page 29.

   Hint: When the import completes, we can start work using the critical data sets.

5. **Enter all remaining MVCs (and any standard Nearline volumes) into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

6. **Start an HSC audit on the panels we filled in Step 5.**

7. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Recovery Site VTSSs.**

8. **Now, and only now, start sending non-critical data to the Recovery Site VTSSs.**

   This completes this procedure, and we’re now up and running again. As soon as the Production Site is back in operation, run, do not walk, to “Business Resumption” on page 41.

Business Continuance - MVC Import

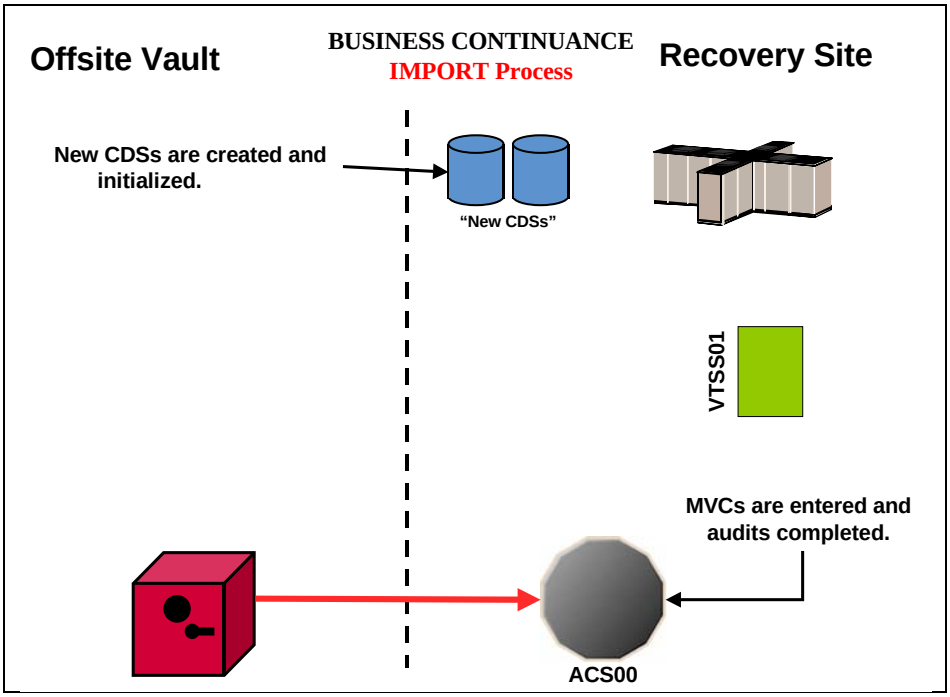


Figure 12. Business Continuance - MVC Import

Business Continuance - VTV Recall

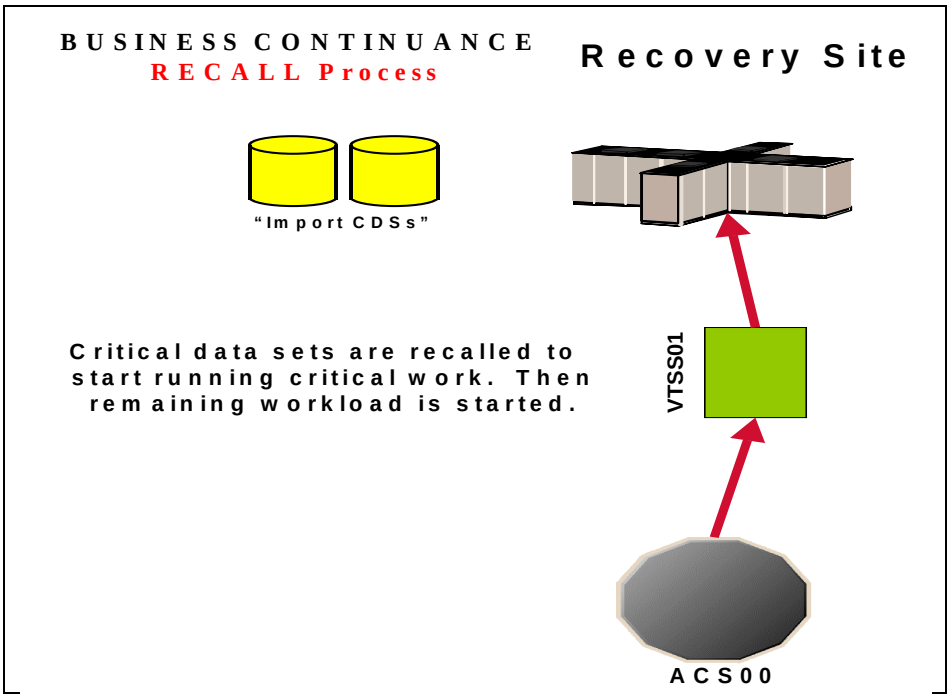


Figure 13. Business Continuance - VTV Recall

Business Continuance - VTV Migration

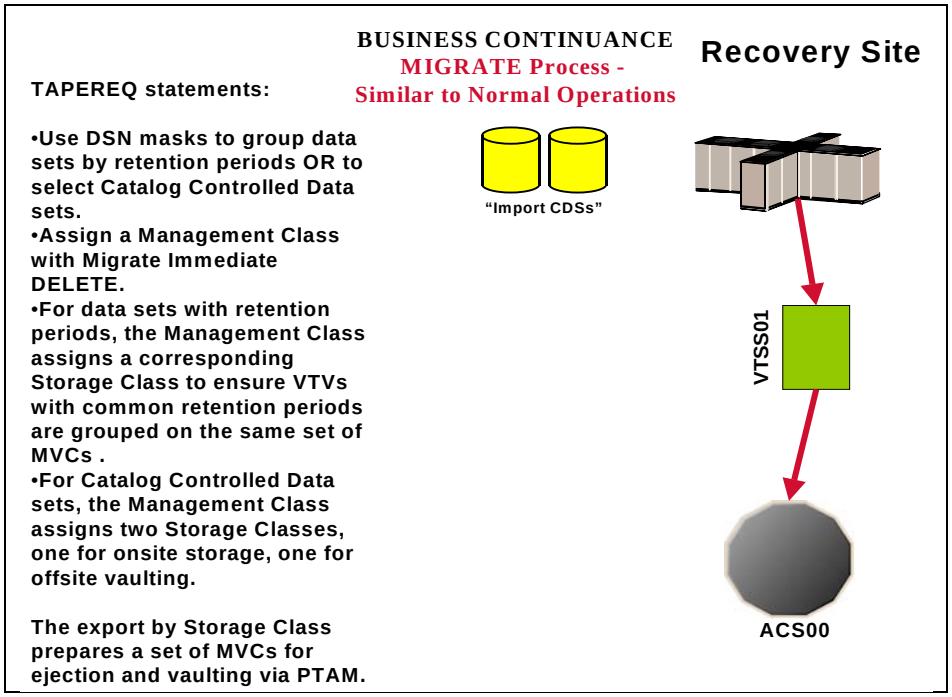


Figure 14. Business Continuance - VTV Migration

Business Resumption

The Production Site is back up as a data center, so it's time to get back to business as usual as shown in Figure 15 on page 42, Figure 16 on page 42, and Figure 17 on page 43.

The switch from Business Continuance to Business Resumption should look familiar, because it's very similar to what we did in "Business Continuance" on page 38...only this time we're using information and resources from the Recovery Site to recreate the Production Site VSM operation.

To resume normal operations:

1. **Clean the Production Site VTSSs.**

   This procedure, which is done by StorageTek hardware service, wipes out any extraneous VTVs that may be lingering around.

2. **Run an Export against all MVC ranges used at Recovery Site, using the Recovery Site CDS.**

Note: At this point, we're done with the Recovery Site...although we might want to keep it up and running until we reach Step 10.

3. **Create a new CDS at the Production Site.**
4. **Enter only critical MVCs into the Production Site ACS.**

   Fill complete rows, one panel at a time.

5. **Start an HSC audit on the panels we filled in Step 4.**
6. **Using the Manifest File from Step 2, run an import into the *new* CDS we created in Step 3.**

Hint: When the import completes, we can start work using the critical data sets.

7. **Enter all remaining MVCs (and any standard Nearline volumes) into the Production Site ACS.**

   Fill complete rows, one panel at a time.

8. **Start an HSC audit on the panels we filled in Step 7.**
9. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Production Site VTSSs.**

Note: After the VTVs that contain critical data sets are VTSS resident, you must enable **access** to the data sets on these VTVs, either by recataloging them, or using JCL statements such as `VOL=SER=vtvnnn`.

10. **Now, and only now, start sending new data to the Production Site VTSSs.**

Business Resumption - VTV Migration

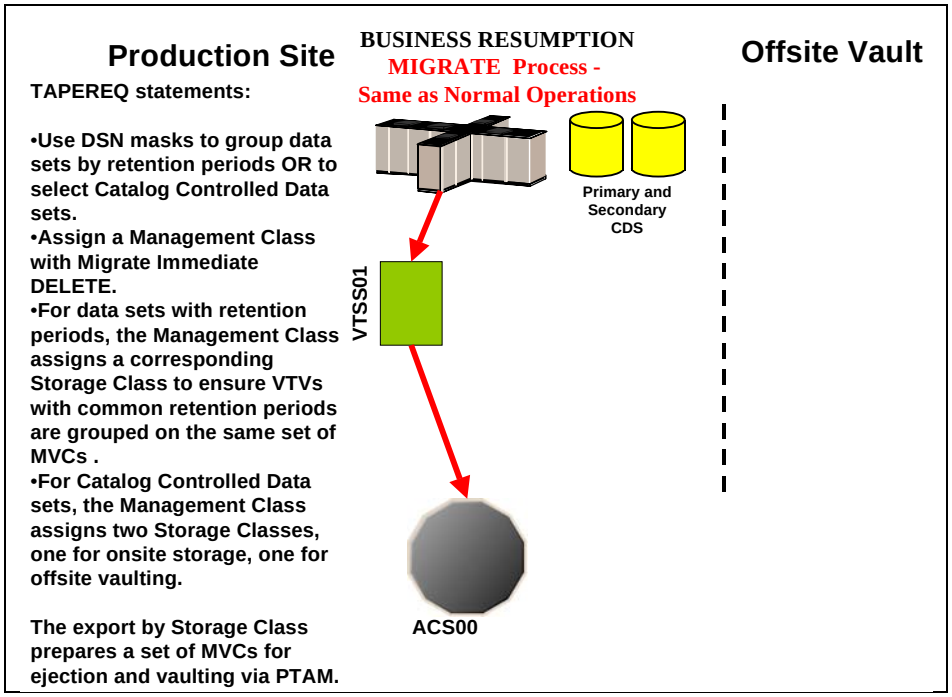


Figure 15. Business Resumption - VTV Migration

Business Resumption - VTV Recall

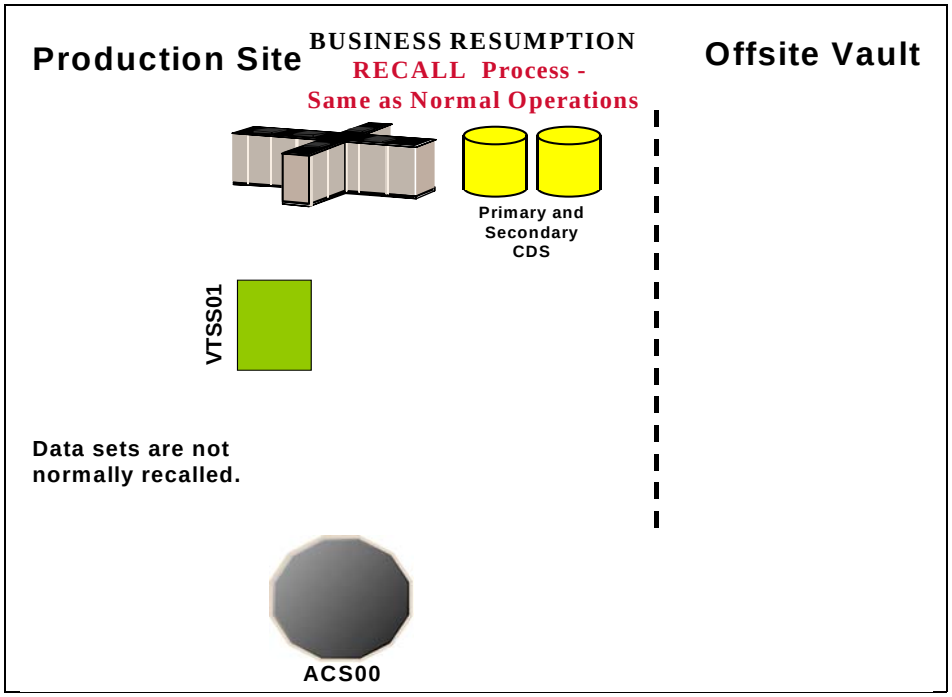


Figure 16. Business Resumption - VTV Recall

Business Resumption - MVC Export

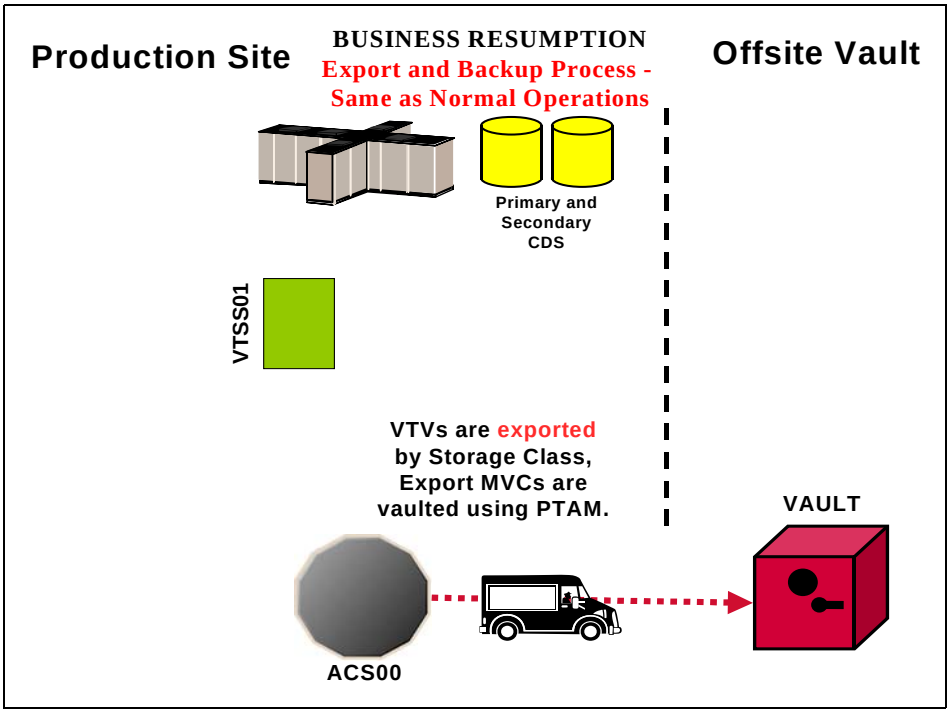


Figure 17. Business Resumption - MVC Export

Offsite Vaulting - The Pickup Truck Access Method for CA-TLMS

This section tells how to use VSM and the PTAM ("Pickup Truck Access Method") to disaster-proof a single-site, offsite vault data processing operation so it can continue and resume operations after a catastrophic failure to the system. Throughout the procedures, you'll see how to do this for data sets with retention periods and for those without retention periods...specifically, they are catalog controlled data sets. The additional sub-flavor is that we're using a TMS to extract the date stamps, and the TMS is CA-TLMS. The advantage of using a TMS is that there are accompanying StorageTek-written utilities that help automate the whole process.

As Figure 18 on page 47 shows, our example configuration consists of an MVS host at the Production Site, attached VTSS, single ACS, and an Offsite Vault. The vaulting of critical data works as follows:

- Data sets **with** retention periods are routed via `TAPEREQs` to VTVs by a series of Management Classes (keyed to retention periods) that specify `MIGRATE IMMEDIATE (DELETE)`. Each Management Class specifies a corresponding Storage Class so that the VTVs for that retention period are grouped on a common set of MVCs. After the migrations complete, you next run the `SWDIMS01` utility to create export MVCs to eject via ExLM for transport to the offsite vault. The VTVs on the export MVCs have expiry dates per the TMS (CA-TLMS). As these VTVs expire, you use an ExLM Custom Volume Report and the `SWDIMS02` utility to identify MVCs with no current VTVs in the offsite vault. You then use CA-TLMS TRS to create a list of these MVCs to return them to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs with a current data generation.
- **Catalog controlled** critical data sets are routed via a `TAPEREQ` to VTVs by a single Management Class that specifies `MIGRATE IMMEDIATE (DELETE)`. This Management Class specifies a two corresponding Storage Classes, one for onsite storage, one for offsite vaulting. After the migrations complete, you next run the `SWDIMS01` utility against the offsite Storage Class to create export MVCs to eject via ExLM for transport to the offsite vault. When ExLM reports MVCs with space utilization of less than a user-specified value, you use the `SWDIMS02` utility to identify these MVCs. You then "logically drain" these MVCs (by draining the onsite MVC), use CA-TLMS TRS to create a pull list of the MVCs eligible for reuse, and then return the offsite MVCs to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs.

Caution: Please note the following gotcha. Some customers use HSC User Exit 06 to enter new tapes into the ACS and define them as scratch. This is exactly what we'd want for Nearline tapes, but scratch MVCs are a **not** a value add. So if you're entering MVCs and HSC User Exit 06 is set to define newly entered tapes as scratch, use the HSC `UEXIT` command to disable the exit before entering tapes to be used as MVCs.

"Offsite Vaulting - The Pickup Truck Access Method for CA-TLMS" consists of the following subsections:

- "Normal Operations" on page 47, where we show what the Production Site looks like when the data is flowing smoothly. Here, we also tell how to set up the system to withstand a failure.
- "Business Continuance" on page 64, which is how to stay open for business if the Production Site takes a major hit.
- "Business Resumption" on page 67, which is how to get back to normal operations once the Production Site is up and running again.

Normal Operations uses the StorageTek VSM Vault Utilities (the `SWDIMS01` and `SWDIMS02` utilities), which you installed in "Installing the VSM Vault Utilities" on page 9.

Normal Operations

On page 45, we gave an overview of the system, disaster recovery needs, and operations to meet those needs. Figure 18 and Figure 19 on page 48 show the system during normal operations for VTV migration and recall. Figure 20 on page 48 shows the export and physical transport to the offsite vault of the MVCs that contain the critical data. There is, as we said, Some Assembly Required, so let's turn to "Migration, Export, Offsite Vault, and MVC Reuse" on page 49 for the details.

Normal Operations - VTV Migration

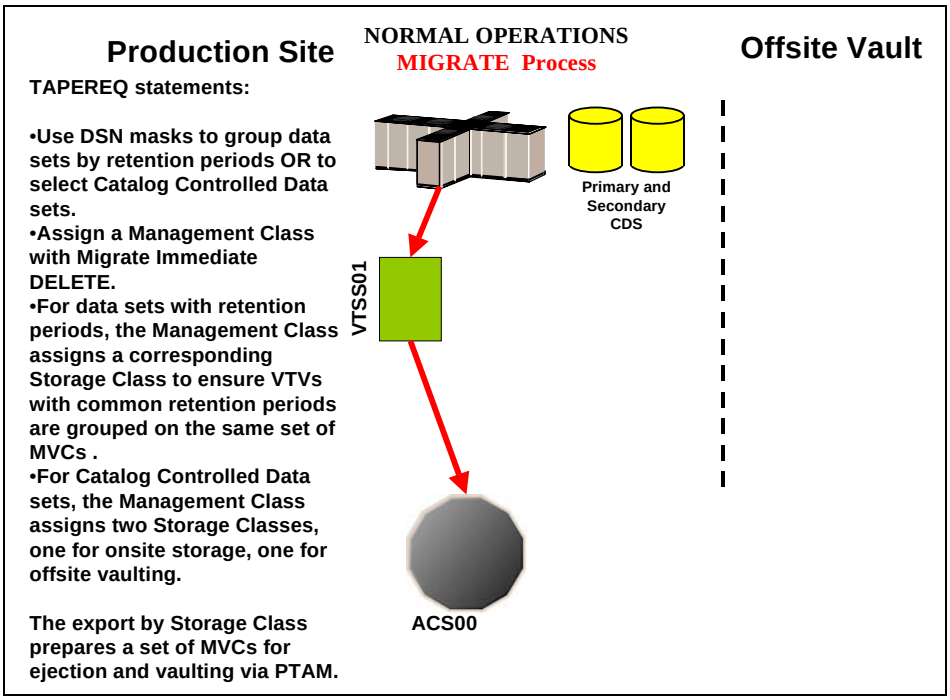


Figure 18. Normal Operations - VTV Migration

Normal Operations - VTV Recall

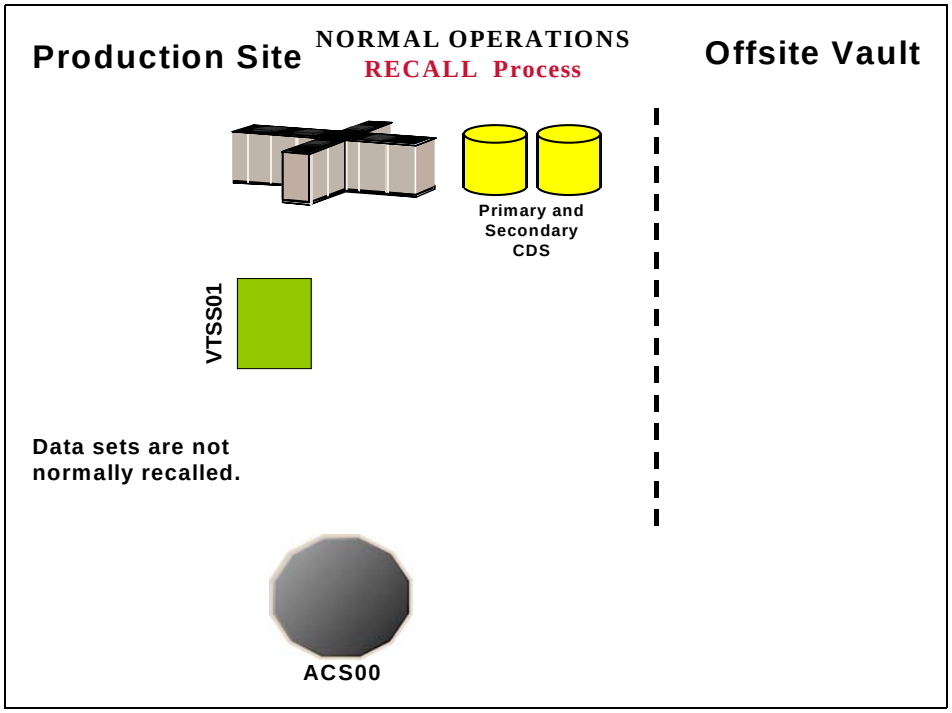


Figure 19. Normal Operations - VTV Recall

Normal Operations - MVC Export

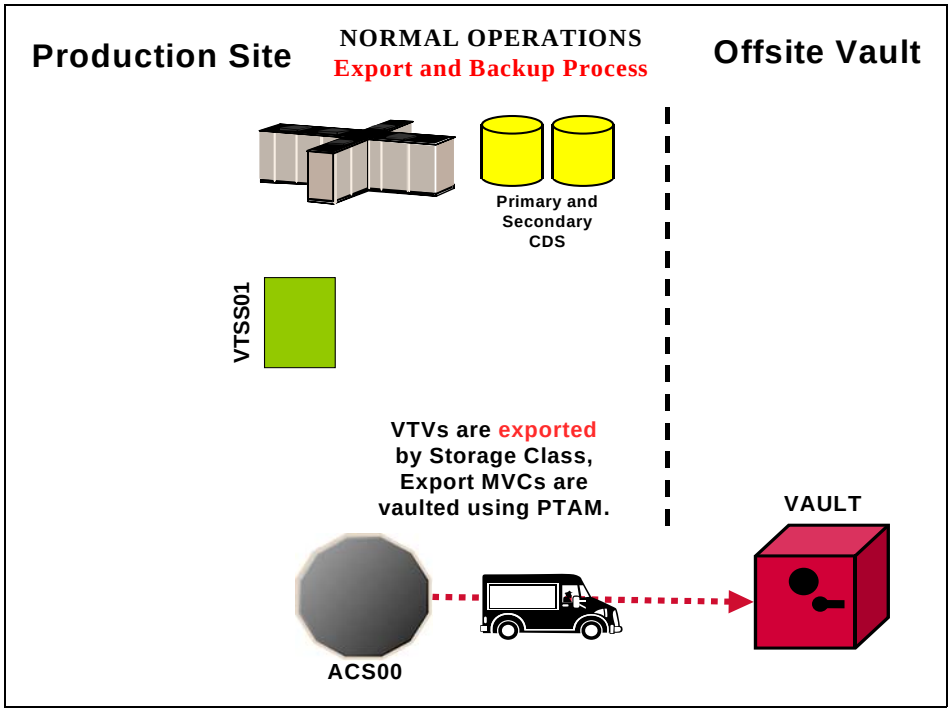


Figure 20. Normal Operations - MVC Export

Migration, Export, Offsite Vault, and MVC Reuse

In “Normal Operations” on page 47, we showed the system under normal operations. This section tells how to route critical data **in data sets with retention periods** and **catalog controlled data sets** to VTVs, migrate them to a common set of MVCs, and vault the MVCs offsite.

Table 13 on page 50 and Table 14 on page 50 are based on the following crucial differences in that data sets that you want to vault:

- First, in Table 13 on page 50, for **data sets with retention periods**, we’re showing a situation where you have multiple data set name masks, each of which corresponds to a different retention period for the data sets selected by that mask. The fewer of these, the better (read what the VTCS documentation says about Too Many Storage Classes). That is, for *each* data set name mask you need a corresponding Management Class and Storage Class.

  With TAPEREQ, in fact, you have lots of choices...for example, you can key on jobname or stepname, if that works better. If you want to key on retention periods, data set name is probably the right move, however. We just want to make sure you know that you **do** have the flexibility to key on different data set groupings with different characteristics.

- In Table 14 on page 50, for **catalog controlled data sets**, we’re again showing the use of a data set name mask on TAPEREQ. As above, use other data set characteristics that you specify on TAPEREQ if that works better. Note that with catalog controlled data sets, the TAPEREQ specifies a single Management Class that duplexes the data via two Storage Classes. This produces an offsite MVC and an onsite MVC with the critical data, which, as we’ll see later, is critical to reusing and recycling MVCs.

So at this point your question is “What do I do if I have both flavors of data sets?” Well, we thought of that, too, because in the real world this is a very likely situation. Back in the MVC reuse section, which is one of the key areas to get right, we break that part of the process down for you. You can read about all of that in “Returning MVCs from the Offsite Vault for Reuse” on page 57. Another useful tool, however, is the overall process flow for
“mixed” shops that we show in “A Sample Offsite Vaulting Work Flow” on page 173.

- In this example, we're selecting the MVCs for offsite vaulting from a Named MVC Pool. This is optional, but it's not a bad idea because:
  - All the non-critical jobs select MVCs either from another Named MVC Pool or from the overall MVC Pool (except for the Named Pool used for offsite vaulting).
  - At the Recovery Site, you only have to define the MVCs in the Named Pool.

  Once again, however, read the *caveats* in the VTCS documentation about the potential gotchas with Named MVC Pools.
- Finally, throughout the procedures, we refer you back to Table 13 or Table 14. In fact, what we *really* mean is your versions of these tables, which you'll have to create to match your shop's situation...as recorded in "Offsite Vaulting Configuration Record".

Table 13. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Data Sets with Retention Periods

Management Class (`mgmt-class-name`)	**Retention Period RETPD(LT,n)**	**Storage Class** (`stor-clas-name`)	**Named MVC Pool** (`poolname`)	**TAPEREQ Data Set Mask** (`mask`)
`mgmtret1`	1 to 7 days	`storret1`	*poolret1*	`maskret1`
		. . .		
`mgmtretn`	Greater than one century	`storretn`	*poolretn*	`maskretn`

Table 14. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Catalog Controlled Data Sets without Retention Periods

Management Class	**Storage Classes**	**Use**	**Named MVC Pool**	**TAPEREQ Data Set Mask**
`mgmtcat`	`storon` `storoff`	On site MVCs Offsite MVCs	`poolcat`	`maskcat`

- Let's not forget the SWDTMS01 and SWDTMS02 utilities, which we'll use in the following sections. For some reference information, see "SWDTMS01 Utility" on page 144 and "SWDTMS02 Utility" on page 155.

  The whole of Normal Operations takes some doing, so let's break it down into the following sections:

  - "Setting Up the System" on page 51
  - "Creating the Export MVCs and Vaulting Them Offsite" on page 55
  - "Returning MVCs from the Offsite Vault for Reuse" on page 57

Setting Up the System

To set up the system:

1. **First, we'll enable the Advanced Management Feature.**

   The Advanced Management Feature is required to implement Storage Classes, run the EXPORT and IMPORT utilities, and use selected Management Class parameters (such as MIGPOL).

2. **Let's build the Management Classes that assign a corresponding Storage Class and do an immediate migrate with delete.**

- **For data sets with retention periods**, *each* of the MGMTclas statements (you create a separate statement for each retention period) looks something like Figure 21:

```
MGMT NAME(mgmtretclass) IMMED(DELETE) MIGPOL(storretclas)DELSCR(YES)
```

Figure 21. Management Classes for VTVs with Critical Data

- **For catalog controlled data sets**, the MGMTclas statement looks something like Figure 22:

```
MGMT NAME(mgmtcat) IMMED(DELETE) MIGPOL(storon,storoff)DELSCR(YES)
```

Figure 22. Management Classes for VTVs with Critical Data - Data Sets Without Retention Periods

In Figure 21 and Figure 22, each Management Class specifies immediate migrate delete, which is designed to immediately put VTVs in this Management Class on the migration queue and delete the VTSS copy once it is migrated. This ensures quick migration and frees the VTSS buffer. We also assign the corresponding Storage Class to the MVCs that contain the migrated VTVs as shown in Table 13. on page 50 or Table 14 on page 50.

Warning: Note that this Management Class specifies DELSCR(YES). If you haven't already done so, **go back and read** the planning information in "DELSCR Considerations" on page 6.

3. **Next, we'll create the Storage Classes that own the MVCs that contain the migrated VTVs.**

   - **For data sets with retention periods**, each of the STORclas statements looks something like Figure 23:

```
STOR NAME(storretname) MVCP(poolret)
```

Figure 23. Storage Classes for Data Sets with Retention Periods

   For the Storage Class names to plug in, see Table 13. on page 50. In Figure 23, the STORclas statement associates each Storage Class with the specified Named MVC Pool.

   - **For catalog controlled data sets**, the STORclas statements look something like Figure 24:

```
STOR NAME(storon) MVCP(poolcat)
STOR NAME(storoff) MVCP(poolcat)
```

Figure 24. Storage Classes for Catalog Controlled Data Sets

   For the Storage Class names to plug in, see Table 14. on page 50. In Figure 24, the STORclas statements associate each Storage Class with the specified Named MVC Pool.

4. **Next, we'll use the MGMTDEF command to load the MGMTclas and STORclas control statements we created in Step 2 on page 51 and Step 3 on page page 52.**

5. **Now let's create TAPEREQ statements to route the critical data to VSM and assign the corresponding Management Class to the data.**

   - **For data sets with retention periods**, *each* of the TAPEREQ statements (you create a separate statement for each retention period) looks something like Figure 25:

```
TAPEREQ DSN(maskretname) MEDIA(VIRTUAL) MGMT(mgmtretname) RETPD(LT,n)
```

Figure 25. TAPEREQ Statements for Data Sets with Retention Periods

Note: The RETPD parameter is not supported under JES3, so use another TAPEREQ data set selection method (for example, JOBNAME).

See Table 13. on page 50 for the correct values to fill in for Figure 25.

   - **For catalog controlled data sets**, the TAPEREQ statement looks something like Figure 26:

```
TAPEREQ DSN(maskcat) MEDIA(VIRTUAL) MGMT(mgmtcat)
```

Figure 26. TAPEREQ Statement for Catalog Controlled Data Sets

   See Table 14. on page 50 for the correct values to fill in for Figure 26.

6. **Use the TREQDEF command to load the TAPEREQ control statements we created in Step 5.**

7. **If they aren't already in place, set up the CA-TLMS location IDs for the exported MVCs.**

   CA-TLMS uses the data set name (and optionally the name of the job that creates the data set) to assign vaulting codes. For more information on setting up vaulting codes to achieve the desired vaulting assignments and rotation, see *CA-TLMS User's Guide*.

   The name of the data set that the SWDIMS01 utility assigns to the MVC in the VMF (Volume Master File) is:

   DSN=prefix.storclas.MVCTAPE.RESERVED

   Where:

   - prefix is the HLQ specified in the MvcDsnPreFiX parameter of the STORclas statement. For more information, see "STORCLAS Statement Example for CA-1" on page 171.
   - storclas is a Storage Class created in Step 3 on page 52.

   For example, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and LE8DAYS for the Storage Class for retention periods of less than 8 days, the SWDIMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.LE8DAYS.MVCTAPE.RESERVED

   Similarly, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and CATOFF for the Storage Class for "offsite" MVCs with catalog controlled data sets, the SWDIMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.CATOFF.MVCTAPE.RESERVED

8. **There is one last item...**

   ...which is defining the MVCs to the TMS. As it says in the VTCS documentation:

   > "Access to the MVCs via an RTD bypasses the MVS intercepts put in place by the tape management system so that it does *not* record within its database any access to the MVCs by VSM and does *not* automatically provide protection against inadvertent overwrites of non-expired data on MVCs. Therefore, if you choose to define MVCs to the tape management system, StorageTek **strongly recommends** that you define them as non-scratch, non-expiring volumes."

   Well, that isn't going to work so well for MVCs that you want to reuse, so we have to find another way. So if you want to define the MVCs to CA-TLMS and adequately protect them, try this:

   a. Do a format extend to add the MVC volsers to the VMF.

   b. Second, define the MVCs as a scratch subpool to CA-TLMS.

   c. Finally, write a rule that says that *only* HSC/VTCS can write to the MVCs as scratches.

   Section 1, Setting Up the System, is done, so let's proceed to "Creating the Export MVCs and Vaulting Them Offsite" on page 55.

To create the export MVCs and vault them offsite:

1. **Ensure that all VTVs with critical data are migrated.**

   The TAPEREQ statements in Step 5 on page 52 are the triggers to start the VTV migrations to the MVCs we'll export. But how do we know when all the VTVs with critical data are migrated? Basically, we write a "watchdog" program that monitors VTV migration and reports when the *last* critical VTV is migrated.

2. **Run the SWDIMS01 utility.**

   The SWDIMS01 utility, working together with the JCL you create to invoke it, does the following:

   - **Runs the EXPORT utility** against the Storage Classes we created in Step 3 on page 52 **and creates a comprehensive Manifest File** that includes all these Storage Classes. We use the STORclas statement of the parameter file to specify one or more Storage Classes. For more information, see "STORCLAS" on page 169.

   Note: If an MVC in a specified Storage Class is in use when the SWDIMS01 utility runs, it will be skipped and will not be exported until the next time the utility runs.

   The comprehensive Manifest File is used, if needed, in Business Continuance mode (see Step 4 on page 64)...so it's probably a good idea to have one copy of the file on disk and another on tape stored offsite.

   - **Creates an MVC update file**, which is used as input to CATINQR in Step 3 on page 56, to set data set names and expiration dates for the MVCs to be sent offsite. These commands set the data set expiration date to 99365 (manual control (permanent)).
   - **Creates a VTV update file**, which can optionally be used as input to CATINQR in Step 3 on page 56, to change the TMS VTV records.

     These control cards reflect the state of the VTVs being sent offsite, and it's probably not a good idea to alter them. If, for some reason, you have a need to change the state of VTVs being sent offsite, contact StorageTek Software Support for information on the use of these cards.
   - Sets the CJOB values to simulate an expiration date...well, for a discussion of all that good stuff, see the following:
     - Step 2 on page 61
     - Step 5 on page 62

Hint: Good Practices suggest that we **only** want to vault volumes that are resident in an ACS. To do this, run the SWDIMS01 utility with the STORCLAS ALL(OFF) parameter of the parameter file. If, however, you want to see a selection list of all volumes that **should** be vaulted, whether they are ACS resident, use the STORCLAS ALL(ON) parameter. For more information, see "STORCLAS" on page 169. **Note that**, even if you specify the ALL(OFF) parameter, the Manifest File is **always** "comprehensive."

For example JCL for the SWDIMS01 utility, see Figure 78 on page 146 and Figure 79 on page 147.

3. **Next, we run CATINQR to update the VMF with MVC information.**

   CATINQR uses the input from Step 2 on page 55 to update the MVC volume records in the VMF with a dummy data set name, expiration date, creation date, and so forth.

Hint: In Step 2, if the SWDIMS01 utility **does not** select any MVCs for update, SWDIMS01 ends with a return code of 4. Because the VMF update in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS01 runs successfully, selects MVCs, and completes with a return code of 0 as shown in Figure 80 on page 148.

4. **To assign the location IDs and rotation to the MVCs, run the CA-TLMS TRS.**

   ...where we set up the location IDs in Step 7 on page 53 and updated the VMF with MVC information in Step 3 on page 56.

   For more information on the CA-TLMS TRS, see *CA-TLMS Administrator and Operator Guide*.

5. **Keying on the Location ID from Step 7 on page 53, we use ExLM to eject the MVCs from the ACS.**

   Figure 81 on page 149 shows example JCL to use ExLM to eject the MVCs.

6. **Finally, we use PTAM (Pickup Truck Access Method) to do the offsite vaulting of the MVCs we ejected in Step 5.**

   Because we have a finite number of MVCs, when VTVs go non-current on exported MVCs, we need a way to cycle these MVCs back into the system for reuse. For that procedure, see "Returning MVCs from the Offsite Vault for Reuse" on page 57.

Returning MVCs from the Offsite Vault for Reuse

In “Creating the Export MVCs and Vaulting Them Offsite” on page 55, we migrated VTVs with critical data to MVCs and exported those MVCs to the Offsite Vault. We don’t have an infinite number of MVCs for vaulting, so we need a way to recycle vaulted MVCs back into the system when some or all VTVs on these MVCs go non-current. And there are three flavors, so pick the one that works for your shop and follow the bouncing ball:

- **Flavor 1 - For sites with *only* data sets with retention periods**, when ExLM reports that all VTVs on an MVC have reached their expiry dates in CA-TLMS, the current VTV count goes to zero and we can return the MVC to the Production Site for reuse. For procedures, see “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 61.
- **Flavor 2 - For sites with *only* catalog controlled data sets**, we use an ExLM report to look for vaulted MVCs that have a usage percentage less than a percentage we specify. In other words, ExLM looks for MVCs that have enough free space to make it worthwhile to reuse these MVCs. What’s a reasonable percentage to specify? The answer, of course, is “it depends on the needs of your shop.” Specify a high percentage and you’ll free up lots of MVCs...at the cost of high VSM/system activity. Specify a low percentage and you’ll have lots of fragmented MVCs in the offsite vault. Try starting with 25% and adjust as needed.

  The way we reuse these MVCs is...well, it’s complex, so let’s slow down and see how it works:

1. MVCs with less than `n` percent usage are marked `LOST`. Now why would I do that? Because I have an equivalent on site copy and I don’t want to put an operator through mounting the offsite MVCs to drain it, see number 2.
2. I try to run a drain against the offsite MVC, it’s lost, so instead VTCS “logically drains” the offsite MVC by recalling all its VTVs from the MVC copy that is onsite. Because the recalled VTVs have the DR Management Class, they are migrated to a new MVC that can be taken offsite. The new MVC is now the input to Step 2 on page 55, which detects MVCs with Storage Classes that require offsite vaulting.

   So VTCS “logically drained” the offsite MVC, which is goodness because the offsite MVC is now 0% full. It is now eligible for selection by Step 1 on page 61 and is returned per Step 8 on page 63.

   And for the step-by-step implementation of this process, see “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 59. **Note that** you are not done until, as it says in Step 4 on page 60, you next complete the procedure described in “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 61...because the logical draining has created MVCs with a zero current VTV count. Now is that cool, or what?

Hint: How often do you want to do this defragmentation? The answer is “it depends on the needs of your shop and your level of vaulting/reuse activity”...but doing this once a week is probably a good starting point.

- **Flavor 3 - For "mixed" sites with *both* catalog controlled data sets *and* data sets with retention periods,** you do the same thing (and for the same reasons) that you do for sites with only catalog controlled data sets. That is, you first do the procedure in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 59, followed by the procedure in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 61.

Note: Each of the three procedures described above **requires** the standard TRS update process described in *CA-TLMS Administrator and Operator Guide*.

Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets.

To Recycle Offsite MVCs That Contain VTVs With Catalog Controlled Data Sets:

1. **First, let's run an ExLM Custom Volume Report...**

   ...to find MVCs with less than a specified percent used.

   Figure 82 on page 157 shows a JCL example that generates an ExLM report that selects vaulted MVCs with less than a specified percent used. In Figure 82 on page 157:

   - `DSN=prefix.storoff.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the VMF, as described in Step 7 on page 53.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "STORCLAS Statement Example for CA-1" on page 171.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specify in Step 1 on page 61. The difference is in the *storclas* value for the middle qualifier on the DSN. In this step, it is the single DSN for the data set assigned to all MVCs that contain VTVs with catalog controlled data sets. In this step, you want to drain *only* the MVCs with catalog controlled data sets, not the MVCs with data sets with retention periods. If you go all the way back to Step 3 on page 52, you'll see that it is, in fact, `storoff`, which is the Storage Class for the offsite (vaulted) MVCs.

   - `loccode` is the location code for the offsite vault.
   - `percent` is the specified percentage use. See page 57 for a discussion of what constitutes a reasonable value.
   - `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 84 on page 159 shows example JCL where the `SWDIMS02` utility:

   - Creates the appropriate `MVCMAINT` commands to mark as `LOST` the MVCs selected in Step 1.
   - Creates the appropriate `MVCDRAIN` commands to make all the swell stuff happen that we described starting in number 1. on page 57.

3. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 59 to enable draining the fragmented offsite MVCs.**

Caution: Here we need to take a look at Figure 84 on page 159. This is the JCL to run the SWDIMS02 utility in Step 2 on page 59. Note that the JCL specifies a DD that contains "simplexed" output cards (SWDSIMP) and a DD that contains "duplexed" output cards (SWDDUP). In this step, you **must** run MVCMAINT against the duplexed cards (SWDDUP)!

Also note that, in Step 2, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

Figure 85 on page 160 shows example JCL to run SWSADMIN against the duplexed cards only if MVCs are selected.

Finally, **Also note that**...we're running a drain to recall the current VTVs from the MVC...remember that these are MVCs that are x percent used. How does this actually happen? For the long version, see page 57. Here's another data point:

> After Step 3 completes, the still current VTV, which had one copy on the offsite MVC which was drained, will have one copy on the onsite MVC, and a new copy on another onsite MVC which has the same Storage Class as the original offsite MVC. In the meantime, the offsite MVC will have had its storage class removed. This new onsite DR MVC will be picked up in the next SWDIMS01 run, and will be sent offsite.

And, additionally:

> The drain process also clears the LOST status of the offsite MVC.

4. **Don't stop now...**

...because, for all the good reasons we gave you back on page 57, this process **feeds into** and **requires** that you next complete the procedure described in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 61.

Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods.

To Recycle Offsite MVCs That Contain VTVs With Data Sets with Retention Periods:

1. **First, let's run an ExLM Custom Volume Report to find any MVCs vaulted for data sets with retention periods with no current VTVs....**

   Figure 83 on page 158 shows a JCL example that generates an ExLM report that selects vaulted MVCs with no current VTVs. In Figure 83 on page 158:

   - `DSN=prefix.storclas.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the VMF, as described in Step 7 on page 53.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "STORCLAS Statement Example for CA-1" on page 171.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specified in Step 1 on page 61. The difference is in the *storclas* value for the middle qualifier on the DSN. In this step, we want to detect all offsite MVCs with 0 VTVs. Therefore, the DSN mask in the ExLM job should be "`prefix.**`".

   - `loccode` is the location code for the offsite vault.

   `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 86 on page 161 shows example JCL where the `SWDIMS02` utility:

   - Creates the TMS `CATINQR` commands to modify TMS volume records for MVCs. These commands set the CJOB value and Expiration Date specified on the `EXPIRYPERIOD` statement to allow a return of the MVC from the vault. For more information, see "EXPIRYPERIOD" on page 167.
   - Creates the appropriate `MVCMAINT` commands to reset the Readonly bit to make these MVCs available for reuse after they're reentered into the ACS.
   - Creates the appropriate `MVCDRAIN` commands to...trust us, it's a good idea, and we'll explain it all in Step 4 on page 62.

3. **Run the CA-TLMS `CATINQR` program to process the control cards generated in Step 2...**

   ...to update the VMF accordingly. We do this, together with some other fun `SWSADMIN` stuff, as shown in the example in Figure 87 on page 162.

4. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 to reset the readonly status of the MVCs that will be returning to the production site...**

   ...and to also do a logical drain on those MVCs. “Now why,” you ask yourself, “Is he telling me to drain an *empty* MVC?” And the answer is that invoking MVCDRAIN does more than just draining VTVs, it also affects an MVC state change, which in this case is a Good Thing. Specifically, the MVDRAIN operation removes the Storage Class from the MVC, which prevents the MVC from being selected when SWDIMS01 is exporting MVCs...and also makes the MVC available to the universe, not just that Storage Class. Don’t worry, these empty MVCs will not have to be mounted to be logically drained.

   Caution: Here we need to take a look at Figure 86 on page 161. This is the JCL to run the SWDIMS02 utility in Step 2 on page 61. Note that the JCL specifies a DD that contains “simplexed” output cards (SWDSIMP) and a DD that contains “duplexed” output cards (SWDDUP). In this step, you **must** run MVCMAINT against the simplexed cards (SWDSIMP)!

   Also note that, in Step 2 on page 61, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in this step will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

   Figure 87 on page 162 shows example JCL to run SWSADMIN against the simplexed cards only if MVCs are selected.

5. **Fasten your seatbelts...**

   ...because we have some TLMS-specific work to do here. With TLMS, unlike CA-1, we can’t use the expiration date to expire a volume if it resides offsite. Uh oh, game over...but wait, there’s a fix. Try this:

   a. Code a TLMS rule that is based on the DSN and the CJOB, which you specified on the CJOB parameter of the EXPIRYPERIOD statement. For more information, see “EXPIRYPERIOD” on page 167.

   b. With SWDIMS01, we set the DSN=prefix.storclas.MVCTAPE.RESERVED and CJOB=vtssname, where vtssname was specified on the VTCS CONFIG VTSS NAME parameter. For example, CJOB-VTSS1.

   c. With SWDIMS02, the DSN stays the same but we can code a second TLMS rule to modify the CJOB parameter for the MVCs selected to return from vault. For example, CJOB=VTSSFREE.

   So, the first coded TLMS rule keys on DSN and CJOB=VTSS1 and sends the MVC offsite (manually controlled). The second TLMS rule keys on DSN and CJOB=VTSSFREE and sends the MVC offsite having a retention period based on the creation date and to expire in n days, which you select. Because the MVC is already offsite, TLMS won’t move the volume but *will* modify its attributes from “manually controlled” to “expire n days after creation”. Shazam...we used CJOB for TLMS to simulate an expiration date.

6. **Run the CA-TLMS TRS batch job(s) to create the list of MVCs to be returned from offsite storage for reuse.**

7. **Using VMS picklists, do the PTAM thing to take the MVCs ejected in Step 5 on page 56 to the offsite vault.**

8. **While you're at the offsite vault...**

   ...pick up any MVCs which are to be moved back to the onsite library. This is a little hard to visualize, so think of it this way: We actually identified these MVCs in Step 2 on page 61 during **a previous run** of this procedure that occurred **greater than** `nn` **days ago**, where nn is the expiry period specified on the `EXPIRYPERIOD` statement. For more information, see "EXPIRYPERIOD" on page 167.

   Great, we're done...that is, until the next time we get to cycle back into the defrag procedure described in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 59.

Business Continuance

Okay, the Unthinkable has occurred, and the Production Site has taken a major hit. Now what? Well, the operations staff next wants to switch to business continuance mode. Note that because of the setup we did in “Normal Operations” on page 47, we can quickly and effectively resume operations because:

- All critical data is migrated and vaulted.
- A Recovery Site (which could be a vendor such as Comdisco) is standing by with VSM installed.

Because we did our homework on the setup, the switch to Business Continuance Mode at the Recovery Site is quick and straightforward via the MVC import shown in Figure 27 on page 65. After we switch to Business Continuance Mode, operations look like Figure 28 on page 65 and Figure 29 on page 66. We’ll use the following procedure to switch to Business Continuance Mode.

To switch to Business Continuance Mode at the Recovery Site, do the following:

1. **Create a new CDS at the Recovery Site.**

   This CDS reflects LIBGEN and VTCS configuration at the Recovery Site; we’ll populate this with VTV and MVC information in Step 4.

2. **Enter only critical MVCs into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

3. **Start an HSC audit on the panels we filled in Step 2.**

4. **Using the *most current* comprehensive Manifest File, run an import into the *new* CDS we created in Step 1.**

   We created the comprehensive Manifest File in Step 2 on page 55.

   Hint: When the import completes, we can start work using the critical data sets.

5. **Enter all remaining MVCs (and any standard Nearline volumes) into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

6. **Start an HSC audit on the panels we filled in Step 5.**

7. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Recovery Site VTSSs.**

8. **Now, and only now, start sending non-critical data to the Recovery Site VTSSs.**

   This completes this procedure, and we’re now up and running again. As soon as the Production Site is back in operation, run, do not walk, to “Business Resumption” on page 67.

Business Continuance - MVC Import

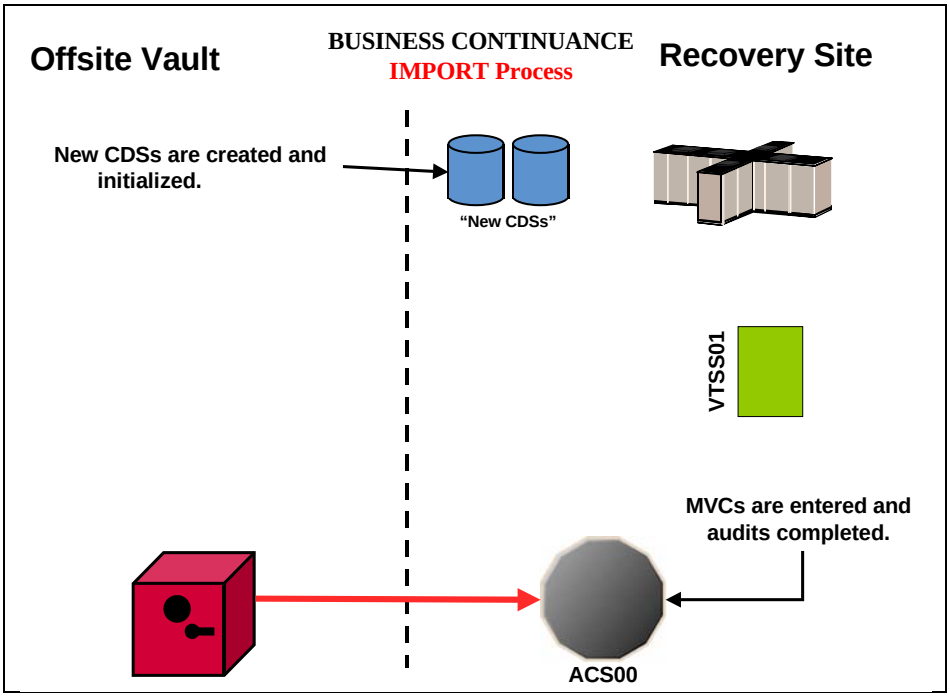


Figure 27. Business Continuance - MVC Import

Business Continuance - VTV Recall

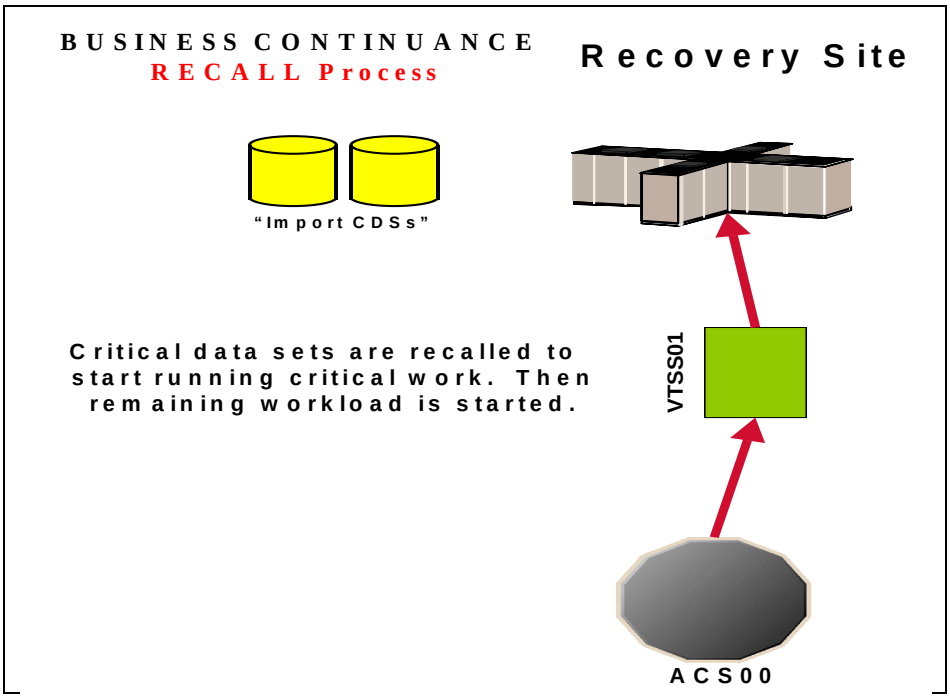


Figure 28. Business Continuance - VTV Recall

Business Continuance - VTV Migration

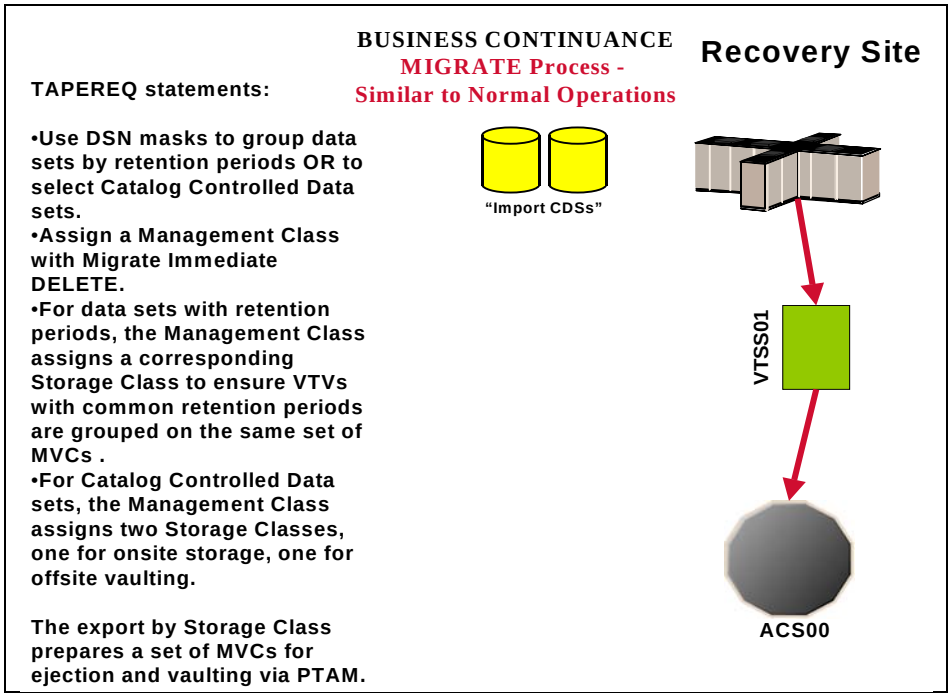


Figure 29. Business Continuance - VTV Migration

Business Resumption

The Production Site is back up as a data center, so it's time to get back to business as usual as shown in Figure 30 on page 68, Figure 31 on page 68, and Figure 32 on page 69.

The switch from Business Continuance to Business Resumption should look familiar, because it's very similar to what we did in "Business Continuance" on page 64...only this time we're using information and resources from the Recovery Site to recreate the Production Site VSM operation.

To resume normal operations:

1. **Clean the Production Site VTSSs.**

   This procedure, which is done by StorageTek hardware service, wipes out any extraneous VTVs that may be lingering around.

2. **Run an Export against all MVC ranges used at Recovery Site, using the Recovery Site CDS.**

   Note: At this point, we're done with the Recovery Site...although we might want to keep it up and running until we reach Step 10.

3. **Create a new CDS at the Production Site.**

4. **Enter only critical MVCs into the Production Site ACS.**

   Fill complete rows, one panel at a time.

5. **Start an HSC audit on the panels we filled in Step 4.**

6. **Using the Manifest File from Step 2, run an import into the *new* CDS we created in Step 3.**

   Hint: When the import completes, we can start work using the critical data sets.

7. **Enter all remaining MVCs (and any standard Nearline volumes) into the Production Site ACS.**

   Fill complete rows, one panel at a time.

8. **Start an HSC audit on the panels we filled in Step 7.**

9. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Production Site VTSSs.**

   Note: After the VTVs that contain critical data sets are VTSS resident, you must enable **access** to the data sets on these VTVs, either by recataloging them, or using JCL statements such as `VOL=SER=vtvnnn`.

10. **Now, and only now, start sending new data to the Production Site VTSSs.**

Business Resumption - VTV Migration

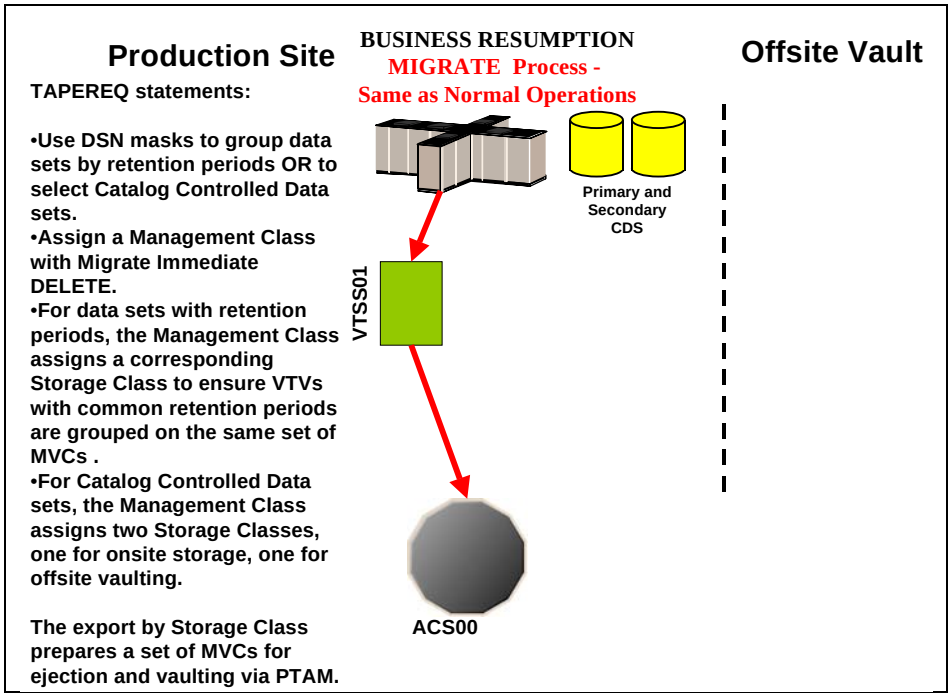


Figure 30. Business Resumption - VTV Migration

Business Resumption - VTV Recall

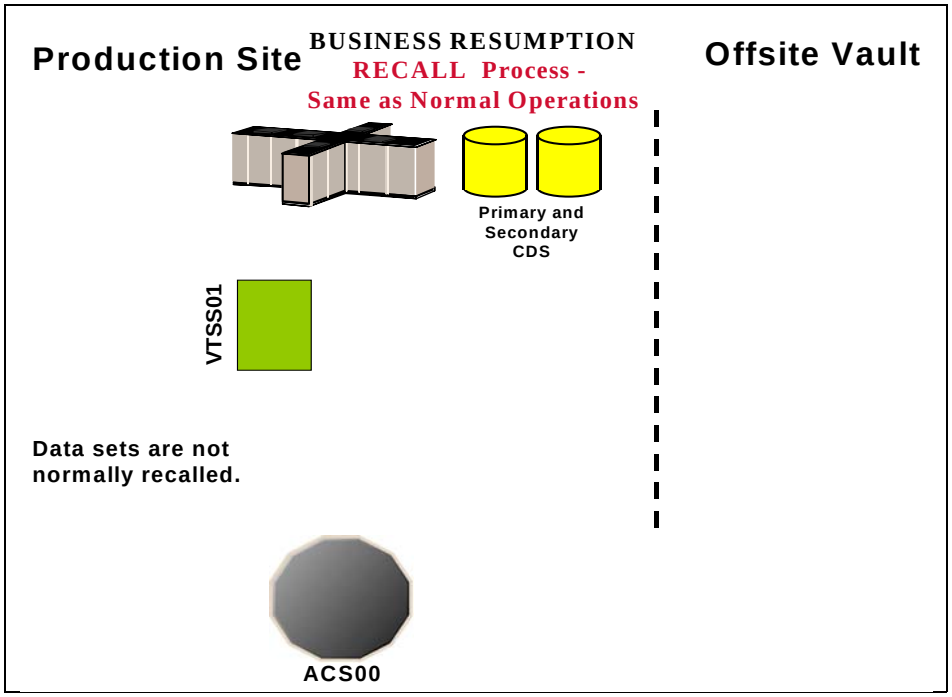


Figure 31. Business Resumption - VTV Recall

Business Resumption - MVC Export

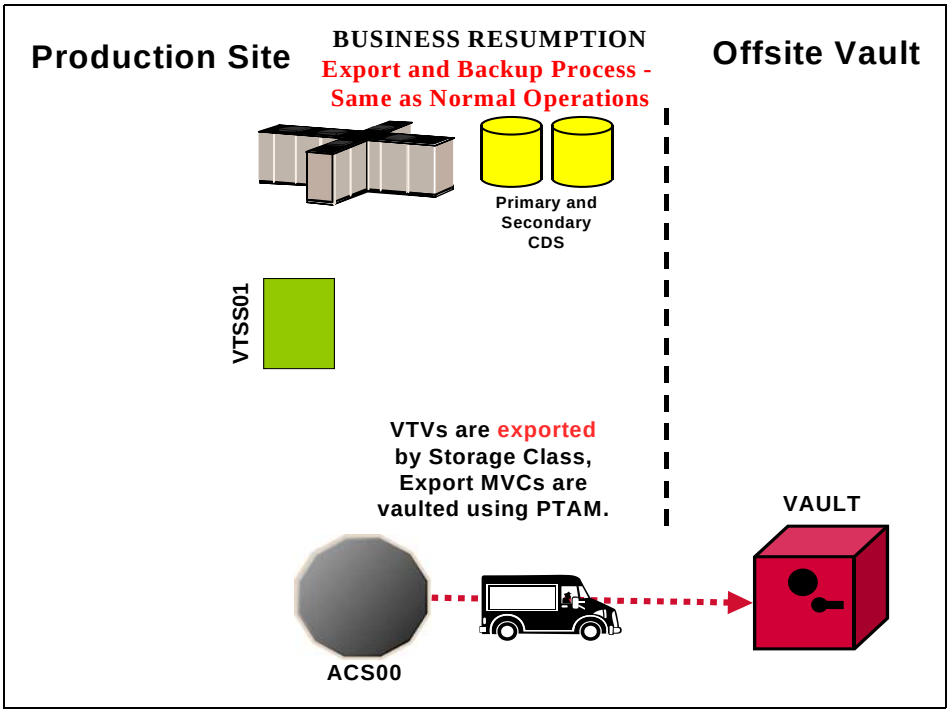


Figure 32. Business Resumption - MVC Export

Offsite Vaulting - The Pickup Truck Access Method for Control-M/Tape

This section tells how to use VSM and the PTAM ("Pickup Truck Access Method") to disaster-proof a single-site, offsite vault data processing operation so it can continue and resume operations after a catastrophic failure to the system. Throughout the procedures, you'll see how to do this for data sets with retention periods and for those without retention periods...specifically, they are catalog controlled data sets. The additional sub-flavor is that we're using a TMS to extract the date stamps, and the TMS is Control-M/Tape. The advantage of using a TMS is that there are accompanying StorageTek-written utilities that help automate the whole process.

As Figure 33 on page 73 shows, our example configuration consists of an MVS host at the Production Site, attached VTSS, single ACS, and an Offsite Vault. The vaulting of critical data works as follows:

- Data sets **with** retention periods are routed via `TAPEREQs` to VTVs by a series of Management Classes (keyed to retention periods) that specify `MIGRATE IMMEDIATE (DELETE)`. Each Management Class specifies a corresponding Storage Class so that the VTVs for that retention period are grouped on a common set of MVCs. After the migrations complete, you next run the `SWDIMS01` utility to create export MVCs to eject via ExLM for transport to the offsite vault. The VTVs on the export MVCs have expiry dates per the TMS (Control-M/Tape). As these VTVs expire, you use an ExLM Custom Volume Report and the `SWDIMS02` utility to identify MVCs with no current VTVs in the offsite vault. You then use the Control-M/Tape `CTTVIM` utility to create a list of these MVCs to return them to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs with a current data generation.
- **Catalog controlled** critical data sets are routed via a `TAPEREQ` to VTVs by a single Management Class that specifies `MIGRATE IMMEDIATE (DELETE)`. This Management Class specifies a two corresponding Storage Classes, one for onsite storage, one for offsite vaulting. After the migrations complete, you next run the `SWDIMS01` utility against the offsite Storage Class to create export MVCs to eject via ExLM for transport to the offsite vault. When ExLM reports MVCs with space utilization of less than a user-specified value, you use the `SWDIMS02` utility to identify these MVCs. You then "logically drain" these MVCs (by draining the onsite MVC), use the Control-M/Tape `CTTVIM utility` to create a pull list of the MVCs eligible for reuse, and then return the offsite MVCs to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs.

Caution: Please note the following gotcha. Some customers use HSC User Exit 06 to enter new tapes into the ACS and define them as scratch. This is exactly what we'd want for Nearline tapes, but scratch MVCs are a **not** a value add. So if you're entering MVCs and HSC User Exit 06 is set to define newly entered tapes as scratch, use the HSC `UEXIT` command to disable the exit before entering tapes to be used as MVCs.

"Offsite Vaulting - The Pickup Truck Access Method for Control-M/Tape" consists of the following subsections:

- "Normal Operations" on page 73, where we show what the Production Site looks like when the data is flowing smoothly. Here, we also tell how to set up the system to withstand a failure.
- "Business Continuance" on page 89, which is how to stay open for business if the Production Site takes a major hit.
- "Business Resumption" on page 92, which is how to get back to normal operations once the Production Site is up and running again.

Normal Operations uses the StorageTek VSM Vault Utilities (the `SWDIMS01` and `SWDIMS02` utilities), so before diving into Normal Operations, let's turn the page and install these utilities.

Normal Operations

On page 71, we gave an overview of the system, disaster recovery needs, and operations to meet those needs. Figure 33 and Figure 34 on page 74 show the system during normal operations for VTV migration and recall. Figure 35 on page 74 shows the export and physical transport to the offsite vault of the MVCs that contain the critical data. There is, as we said, Some Assembly Required, so let's turn to "Migration, Export, Offsite Vault, and MVC Reuse" on page 75 for the details.

Normal Operations - VTV Migration

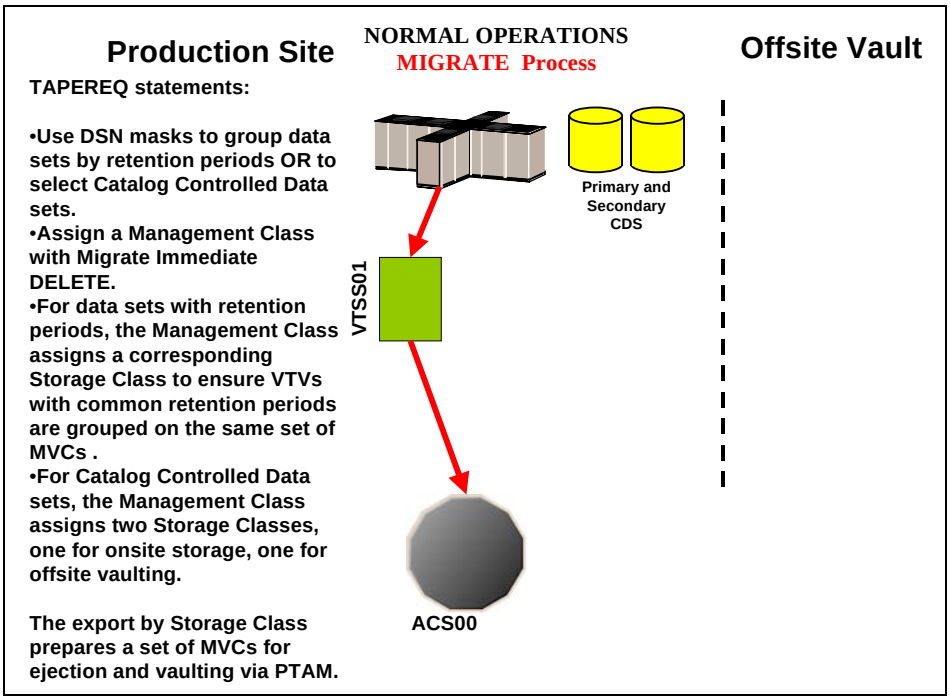


Figure 33. Normal Operations - VTV Migration

Normal Operations - VTV Recall

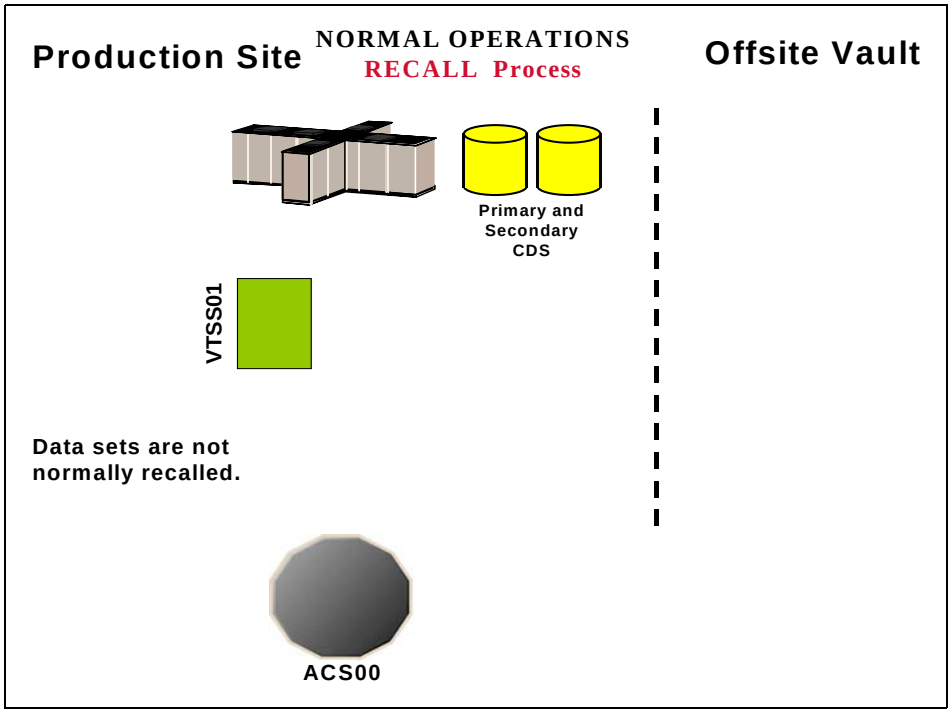


Figure 34. Normal Operations - VTV Recall

Normal Operations - MVC Export

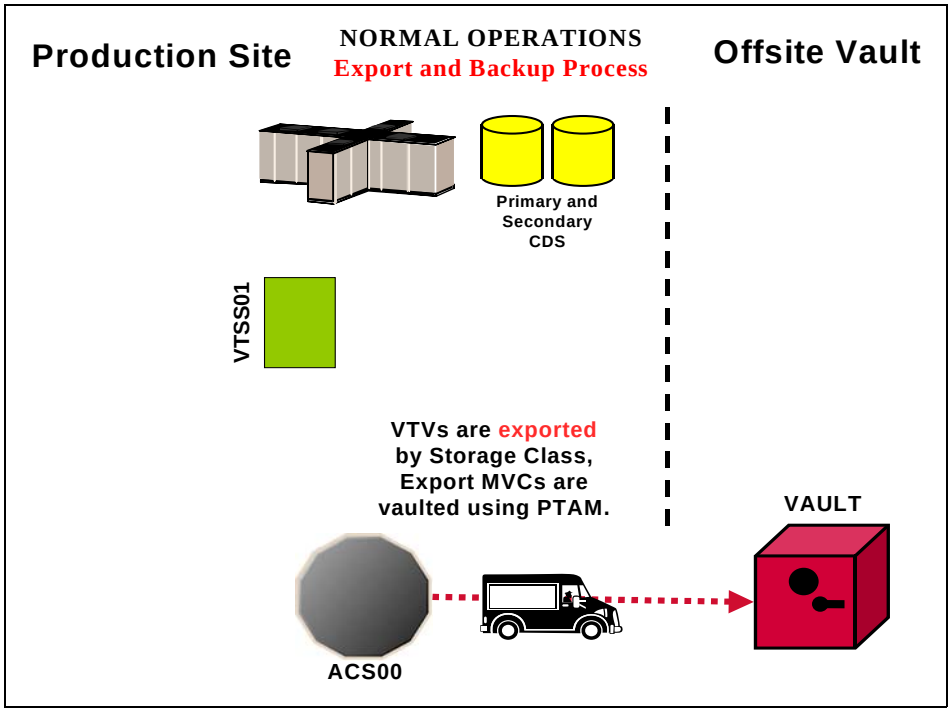


Figure 35. Normal Operations - MVC Export

Migration, Export, Offsite Vault, and MVC Reuse

In “Normal Operations” on page 73, we showed the system under normal operations. This section tells how to route critical data **in data sets with retention periods** and **catalog controlled data sets** to VTVs, migrate them to a common set of MVCs, and vault the MVCs offsite.

Table 16 on page 76 and Table 15 on page 76 are based on the following crucial differences in that data sets that you want to vault:

- First, in Table 15 on page 76, for **catalog controlled data sets**, we’re again showing the use of a data set name mask on TAPEREQ. As above, use other data set characteristics that you specify on TAPEREQ if that works better. Note that with catalog controlled data sets, the TAPEREQ specifies a single Management Class that duplexes the data via two Storage Classes. This produces an offsite MVC and an onsite MVC with the critical data, which, as we’ll see later, is critical to reusing and recycling MVCs.
- In Table 16 on page 76, for **data sets with retention periods**, we’re showing a situation where you have multiple data set name masks, each of which corresponds to a different retention period for the data sets selected by that mask. The fewer of these, the better (read what the VTCS documentation says about Too Many Storage Classes). That is, for *each* data set name mask you need a corresponding Management Class and Storage Class.

  With TAPEREQ, in fact, you have lots of choices...for example, you can key on jobname or stepname, if that works better. If you want to key on retention periods, data set name is probably the right move, however. We just want to make sure you know that you **do** have the flexibility to key on different data set groupings with different characteristics.

So at this point your question is “What do I do if I hav***e both*** flavors of data sets?” Well, we thought of that, too, because in the real world this is a very likely situation. Back in the MVC reuse section, which is one of the key areas to get right, we break that part of the process down for you. You can read about all of that in “Returning MVCs from the Offsite Vault for Reuse” on page 83. Another useful tool, however, is the overall process flow for “mixed” shops that we show in “A Sample Offsite Vaulting Work Flow” on page 173.

- In this example, we're selecting the MVCs for offsite vaulting from a Named MVC Pool. This is optional, but it's not a bad idea because:
  - All the non-critical jobs select MVCs either from another Named MVC Pool or from the overall MVC Pool (except for the Named Pool used for offsite vaulting).
  - At the Recovery Site, you only have to define the MVCs in the Named Pool.

  Once again, however, read the *caveats* in the VTCS documentation about the potential gotchas with Named MVC Pools.
- Finally, throughout the procedures, we refer you back to Table 16 or Table 15. In fact, what we *really* mean is your versions of these tables, which you'll have to create to match your shop's situation...as recorded in "Offsite Vaulting Configuration Record" on page 177.

Table 15. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Catalog Controlled Data Sets without Retention Periods

Management Class	Storage Classes	Use	Named MVC Pool	TAPEREQ Data Set Mask
mgmtcat	storon storoff	On site MVCs Offsite MVCs	poolcat	maskcat

Table 16. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Data Sets with Retention Periods

Management Class (mgmt-class-name)	Retention Period RETPD(LT,n)	Storage Class (stor-clas-name)	Named MVC Pool (poolname)	TAPEREQ Data Set Mask (mask)
mgmtret1	1 to 7 days	storret1	*poolret1*	maskret1
		. . .		
mgmtretn	Greater than one century	storretn	*poolretn*	maskretn

- Let's not forget the SWDTMS01 and SWDTMS02 utilities, which we'll use in the following sections. For some reference information, see "SWDTMS01 Utility" on page 144 and "SWDTMS02 Utility" on page 155.

  The whole of Normal Operations takes some doing, so let's break it down into the following sections:

  - "Setting Up the System" on page 77
  - "Creating the Export MVCs and Vaulting Them Offsite" on page 81
  - "Returning MVCs from the Offsite Vault for Reuse" on page 83

Setting Up the System

To set up the system:

1. **First, we'll enable the Advanced Management Feature.**

   The Advanced Management Feature is required to implement Storage Classes, run the EXPORT and IMPORT utilities, and use selected Management Class parameters (such as MIGPOL).

2. **Let's build the Management Classes that assign a corresponding Storage Class and do an immediate migrate with delete.**

   - **For data sets with retention periods**, *each* of the MGMTclas statements (you create a separate statement for each retention period) looks something like Figure 36:

```
MGMT NAME(mgmtretn) IMMED(DELETE) MIGPOL(storretn) DELSCR(YES)
```

Figure 36. Management Classes for VTVs with Critical Data

   - **For catalog controlled data sets**, the MGMTclas statement looks something like Figure 37:

```
MGMT NAME(mgmtcat) IMMED(DELETE) MIGPOL(storon,storoff) DELSCR(YES)
```

Figure 37. Management Classes for VTVs with Critical Data - Catalog Controlled Data Sets

   In Figure 36 and Figure 37, each Management Class specifies immediate migrate delete, which is designed to immediately put VTVs in this Management Class on the migration queue and delete the VTSS copy once it is migrated. This ensures quick migration and frees the VTSS buffer. We also assign the corresponding Storage Class to the MVCs that contain the migrated VTVs as shown in Table 16. on page 76 or Table 15 on page 76.

Warning: Note that this Management Class specifies DELSCR(YES). If you haven't already done so, **go back and read** the planning information in "DELSCR Considerations" on page 6.

3. **Next, we'll create the Storage Classes that own the MVCs that contain the migrated VTVs.**

   - **For data sets with retention periods**, each of the STORclas statements (you create a separate statement for each retention period) looks something like Figure 38:

```
STOR NAME(storretn) MVCP(poolret)
```

Figure 38. Storage Classes for Data Sets with Retention Periods

   For the Storage Class names to plug in, see Table 16. on page 76. In Figure 38, the STORclas statement associates each Storage Class with the specified Named MVC Pool.

   - **For catalog controlled data sets**, the STORclas statements looks something like Figure 39:

```
STOR NAME(storon) MVCP(poolcat)
STOR NAME(storoff) MVCP(poolcat)
```

Figure 39. Storage Classes for Catalog Controlled Data Sets

   For the Storage Class names to plug in, see Table 15. on page 76. In Figure 39, the STORclas statements associate each Storage Class with the specified Named MVC Pool.

4. **Next, we'll use the MGMTDEF command to load the MGMTclas and STORclas control statements we created in Step 2 on page 77 and Step 3 on page page 78.**

5. **Now let's create TAPEREQ statements to route the critical data to VSM and assign the corresponding Management Class to the data.**

   - **For data sets with retention periods**, each of the TAPEREQ statements (you create a separate statement for each retention period) looks something like Figure 40:

```
TAPEREQ DSN(maskretn) MEDIA(VIRTUAL) MGMT(mgmtretn) RETPD(LT,n)
```

Figure 40. TAPEREQ Statements for Data Sets with Retention Periods

Note: The RETPD parameter is not supported under JES3, so use another TAPEREQ data set selection method (for example, JOBNAME).

   See Table 16. on page 76 for the correct values to fill in for Figure 40.

   - **For catalog controlled data sets**, the TAPEREQ statement looks something like Figure 41:

```
TAPEREQ DSN(maskcat) MEDIA(VIRTUAL) MGMT(mgmtcat)
```

Figure 41. TAPEREQ Statement for Catalog Controlled Data Sets

   See Table 15. on page 76 for the correct values to fill in for Figure 41.

6. **Use the TREQDEF command to load the TAPEREQ control statements we created in Step 5.**

7. **If they aren't already in place, set up the Control-M/Tape vault codes for the exported MVCs.**

   Control-M/Tape can use among other things the data set name and jobname to assign vault codes. These two values will be set for an MVC by the SWDTMS01 utility so any vaulting rules should use one or both of these values. For more information on setting up vaulting codes to achieve the desired vaulting assignments and rotation, see *Control-M/Tape User Guide*.

   The name of the data set that the SWDTMS01 utility assigns to the MVCs in the Media Database is:

   DSN=prefix.storclas.MVCTAPE.RESERVED

   Where:

   - prefix is the HLQ specified in the MvcDsnPreFiX parameter of the STORclas statement. For more information, see "STORCLAS Statement Example for Control-M/Tape" on page 171.
   - storclas is a Storage Class created in Step 3 on page 78.

   For example, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and LE8DAYS for the Storage Class for retention periods of less than 8 days, the SWDTMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.LE8DAYS.MVCTAPE.RESERVED

   Similarly, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and CATOFF for the Storage Class for "offsite" MVCs with catalog controlled data sets, the SWDTMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.CATOFF.MVCTAPE.RESERVED

   The jobname assigned to the MVC by the SWDTMS01 utility is the VTSS name where the MVC was last used. This value is obtained from an MVC report.

The following is an example of a rule that will cause MVCs to be vaulted.

```
ON DATASET=prefix.storclass.MVCTAPE.RESERVED
DO RETENTION=RET FROM VAULT
DO VAULT=vault name
UNTIL PERMANENT
```

where '`vault name`' is where you wish to put the MVCs.

If there is more than one storage class being sent offsite, then the data set name could be specified using generic characters, e.g. `prefix.*.MVCTAPE.RESERVED`. Other ON selection parts could be included in the rule if required including ON JOBNAME however the two DO parts must be coded as is. For further information, see *Control-M/Tape Users Guide*.

8. There is one last item...

...which is defining the MVCs to the TMS. As it says in the VTCS documentation:

> "Access to the MVCs via an RTD bypasses the MVS intercepts put in place by the tape management system so that it does *not* record within its database any access to the MVCs by VSM and does *not* automatically provide protection against inadvertent overwrites of non-expired data on MVCs. Therefore, if you choose to define MVCs to the tape management system, StorageTek **strongly recommends** that you define them as non-scratch, non-expiring volumes."

Well, that isn't going to work so well for MVCs that you want to reuse, so we have to find another way. So if you want to define the MVCs to Control-M/Tape and adequately protect them, try this:

a. Do a format extend to add the MVC volsers to the Media Database.

b. Second, define the MVCs as a scratch pool to Control-M/Tape.

c. Finally, write a rule that says that *only* HSC/VTCS can write to the MVCs as scratches.

Section 1, Setting Up the System, is done, so let's proceed to "Creating the Export MVCs and Vaulting Them Offsite" on page 81.

Creating the Export MVCs and Vaulting Them Offsite

To create the export MVCs and vault them offsite:

1. **Ensure that all VTVs with critical data are migrated.**

   The TAPEREQ statements in Step 5 on page 78 are the triggers to start the VTV migrations to the MVCs we'll export. But how do we know when all the VTVs with critical data are migrated? Basically, we write a "watchdog" program that monitors VTV migration and reports when the *last* critical VTV is migrated.

2. **Ensure that you specify a value for the CTVoLSNuM parameter of the STORCLAS statement.**

   For more information, see Figure 92. on page 171.

3. **Run the SWDIMS01 utility.**

   The SWDIMS01 utility, working together with the JCL you create to invoke it, does the following:

   - **Runs the EXPORT utility** against **all** Storage Classes we created in Step 3 on page 78 **and creates a comprehensive Manifest File** that includes all these Storage Classes. We use the STORclas statement of the parameter file to specify one or more Storage Classes. For more information, see "STORCLAS Statement Example for Control-M/Tape" on page 171.

   Note: If an MVC in a specified Storage Class is in use when the SWDIMS01 utility runs, it will be skipped and will not be exported until the next time the utility runs.

   The comprehensive Manifest File is used, if needed, in Business Continuance mode (see Step 4 on page 89)...so it's probably a good idea to have one copy of the file on disk and another on tape stored offsite.

   - **Creates an MVC update file**, which is used as input to CTIMUP in Step 4 on page 82, to set data set names and expiration dates for the MVCs to be sent offsite. These commands set the data set expiration date to PERMANENT.
   - **Creates a VTV update file**, which can optionally be used as input to CTIMUP in Step 4 on page 82, to change the TMS VTV records.

     These control cards reflect the state of the VTVs being sent offsite, and it's probably not a good idea to alter them. If, for some reason, you have a need to change the state of VTVs being sent offsite, contact StorageTek Software Support for information on the use of these cards.

Hint: Good Practices suggest that we **only** want to vault volumes that are resident in an ACS. To do this, run the SWDIMS01 utility with the STORCLAS ALL(OFF) parameter of the parameter file. If, however, you want to see a selection list of all volumes that **should** be vaulted, whether they are ACS resident, use the STORCLAS ALL(ON) parameter. For more information, see "STORCLAS Statement Example for Control-M/Tape" on page 171. **Note that**, even if you specify the ALL(OFF) parameter, the Manifest File is **always** "comprehensive."

For example JCL for the SWDIMS01 utility, see Figure 78 on page 146 and Figure 79 on page 147.

4. **Next, we run CTIMUP to update the Media Database with MVC information.**

   CTIMUP uses the input from Step 3 on page 81 to update the MVC volume records in the Media Database with a dummy data set name, expiration date, creation date, and so forth.

Hint: In Step 3, if the SWDIMS01 utility **does not** select any MVCs for update, SWDIMS01 ends with a return code of 4. Because the Media Database update in Step 4 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS01 runs successfully, selects MVCs, and completes with a return code of 0 as shown in Figure 80 on page 148.

5. **To assign the vault codes and rotation to the MVCs, run the Control-M/Tape CTTVIM utility.**

   ...where we set up the vault codes in Step 7 on page 79 and updated the Media Database with MVC information in Step 4 on page 82.

   For more information on the Control-M/Tape CTTVIM, see *INCONTROL Utility Guide*.

6. **Keying on the Vault Code from Step 7 on page 79, we use ExLM to eject the MVCs from the ACS.**

   Figure 81 on page 149 shows example JCL to use ExLM to eject the MVCs.

7. **Finally, we use PTAM (Pickup Truck Access Method) to do the offsite vaulting of the MVCs we ejected in Step 6.**

   Because we have a finite number of MVCs, when VTVs go non-current on exported MVCs, we need a way to cycle these MVCs back into the system for reuse. For that procedure, see "Returning MVCs from the Offsite Vault for Reuse" on page 83.

Returning MVCs from the Offsite Vault for Reuse

In “Creating the Export MVCs and Vaulting Them Offsite” on page 81, we migrated VTVs with critical data to MVCs and exported those MVCs to the Offsite Vault. We don’t have an infinite number of MVCs for vaulting, so we need a way to recycle vaulted MVCs back into the system when some or all VTVs on these MVCs go non-current. And there are three flavors, so pick the one that works for your shop and follow the referenced procedure:

1. **Flavor 1 - Sites with *only* data sets with retention periods**, when ExLM reports that all VTVs on an MVC have reached their expiry dates in Control-M/Tape, the current VTV count goes to zero and we can return the MVC to the Production Site for reuse. For procedures, see “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 87.

2. **Flavor 2 - For sites with *only* catalog controlled data sets**, we use an ExLM report to look for vaulted MVCs that have a usage percentage less than a percentage we specify. In other words, ExLM looks for MVCs that have enough free space to make it worthwhile to reuse these MVCs. What’s a reasonable percentage to specify? The answer, of course, is “it depends on the needs of your shop.” Specify a high percentage and you’ll free up lots of MVCs...at the cost of high VSM/system activity. Specify a low percentage and you’ll have lots of fragmented MVCs in the offsite vault. Try starting with 25% and adjust as needed.

   The way we reuse these MVCs is...well, it’s complex, so let’s slow down and see how it works:

   1. MVCs with less than `n` percent usage are marked `LOST`. Now why would I do that? Because I have an equivalent on site copy and I don’t want to put an operator through mounting the offsite MVCs to drain it, see number 2.

   2. I try to run a drain against the offsite MVC, it’s lost, so instead VTCS “logically drains” the offsite MVC by recalling all its VTVs from the MVC copy that is onsite. Because the recalled VTVs have the DR Management Class, they are migrated to a new MVC that can be taken offsite. The new MVC is now the input to Step 3 on page 81, which detects MVCs with Storage Classes that require offsite vaulting.

      So VTCS “logically drained” the offsite MVC, which is goodness because the offsite MVC is now 0% full. It is now eligible for selection by Step 1 on page 87 and is returned per Step 7 on page 88.

   And for the step-by-step implementation of this process, see “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 85.

Note that you are not done until, as it says in Step 4 on page 86, you next complete the procedure described in “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 87...because the logical draining has created MVCs with a zero current VTV count. Is that cool, or what?

Hint: How often do you want to do this defragmentation? The answer is “it depends on the needs of your shop and your level of vaulting/reuse activity”...but doing this once a week is probably a good starting point.

3. **Flavor 3 - For "mixed" sites with *both* catalog controlled data sets *and* data sets with retention periods,** you do the same thing (and for the same reasons) that you do for sites with only catalog controlled data sets. That is, you first do the procedure in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 85, followed by the procedure in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 87.

Note: Each of the three procedures described above **requires** the standard `CTTVTM` update process described in *INCONTROL Utility Guide*.

Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets.

To Recycle Offsite MVCs That Contain VTVs With Catalog Controlled Data Sets:

1. **First, let's run an ExLM Custom Volume Report...**

   ...to find MVCs with less than a specified percent used.

   Figure 82 on page 157 shows a JCL example that generates an ExLM report that selects vaulted MVCs with less than a specified percent used. In Figure 82 on page 157:

   - `DSN=prefix.storoff.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the Media Database, as described in Step 7 on page 79.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "STORCLAS Statement Example for CA-1" on page 171.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specify in Step 1 on page 87. The difference is in the *storclas* value for the middle qualifier on the DSN. In this step, it is the single DSN for the data set assigned to all MVCs that contain VTVs with catalog controlled data sets. In this step, you want to drain ***only*** the MVCs with catalog controlled data sets, not the MVCs with data sets with retention periods. If you go all the way back to Step 3 on page 78, you'll see that it is, in fact, `storoff`, which is the Storage Class for the offsite (vaulted) MVCs.

   - `loccode` is the location code for the offsite vault.
   - `percent` is the specified percentage use. See page 83 for a discussion of what constitutes a reasonable value.
   - `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 84 on page 159 shows example JCL where the `SWDIMS02` utility:

   - Creates the appropriate `MVCMAINT` commands to mark as `LOST` the MVCs selected in Step 1.
   - Creates the appropriate `MVCDRAIN` commands to make all the swell stuff happen that we described starting in number 1. on page 83.

3. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 85 to enable draining the fragmented offsite MVCs.**

Caution: Here we need to take a look at Figure 84 on page 159. This is the JCL to run the SWDIMS02 utility in Step 2 on page 85. Note that the JCL specifies a DD that contains "simplexed" output cards (SWDSIMP) and a DD that contains "duplexed" output cards (SWDDUP). In this step, you **must** run MVCMAINT against the duplexed cards (SWDDUP)!

Also note that, in Step 2, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

Figure 85 on page 160 shows example JCL to run SWSADMIN against the duplexed cards only if MVCs are selected.

Finally, **Also note that**...we're running a drain to recall the current VTVs from the MVC...remember that these are MVCs that are x percent used. How does this actually happen? For the long version, see page 83. Here's another data point:

> After Step 3 completes, the still current VTV, which had one copy on the offsite MVC which was drained, will have one copy on the onsite MVC, and a new copy on another onsite MVC which has the same Storage Class as the original offsite MVC. In the meantime, the offsite MVC will have had its storage class removed. This new onsite DR MVC will be picked up in the next SWDIMS01 run, and will be sent offsite.

And, additionally:

> The drain process also clears the LOST status of the offsite MVC.

4. **Don't stop now...**

...because, for all the good reasons we gave you back on page 83, this process **feeds into** and **requires** that you next complete the procedure described in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 87.

Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods.

To Recycle Offsite MVCs That Contain VTVs With Data Sets with Retention Periods:

1. **First, let's run an ExLM Custom Volume Report to find any MVCs vaulted for data sets with retention periods with no current VTVs....**

   Figure 83 on page 158 shows a JCL example that generates an ExLM report that selects vaulted MVCs with no current VTVs. In Figure 83 on page 158:

   - `DSN=prefix.storclas.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the Media Database, as described in Step 7 on page 79.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "STORCLAS Statement Example for CA-1" on page 171.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specified in Step 1 on page 85. The difference is in the *storclas* value for the Second Level Qualifier on the DSN. In this step, we want to detect all offsite MVCs with 0 VTVs. Therefore, the DSN mask in the ExLM job should be "`prefix.**`".

   - `loccode` is the location code for the offsite vault.

   `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 86 on page 161 shows example JCL where the `SWDIMS02` utility:

   - Creates the TMS `CTIMUP` commands to modify TMS volume records for MVCs. These commands set the Expiration Date specified on the `EXPIRYPERIOD` statement to allow a return of the MVC from the vault. For more information, see "EXPIRYPERIOD" on page 167.
   - Creates the appropriate `MVCMAINT` commands to reset the Readonly bit to make these MVCs available for reuse after they're reentered into the ACS.
   - Creates the appropriate `MVCDRAIN` commands to...trust us, it's a good idea, and we'll explain it all in Step 4 on page 88.

3. **Run the Control-M/Tape `CTIMUP` program to process the control cards generated in Step 2...**

   ...to update the Media Database accordingly. We do this, together with some other fun `SWSADMIN` stuff, as shown in the example in Figure 87 on page 162.

4. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 87 to reset the readonly status of the MVCs that will be returning to the production site...**

   ...and to also do a logical drain on those MVCs. “Now why,” you ask yourself, “Is he telling me to drain an *empty* MVC?” And the answer is that invoking MVCDRAIN does more than just draining VTVs, it also affects an MVC state change, which in this case is a Good Thing. Specifically, the MVDRAIN operation removes the Storage Class from the MVC, which prevents the MVC from being selected when SWDIMS01 is exporting MVCs...and also makes the MVC available to the universe, not just that Storage Class. Don’t worry, these empty MVCs will not have to be mounted to be logically drained.

   Caution: Here we need to take a look at Figure 86 on page 161. This is the JCL to run the SWDIMS02 utility in Step 2 on page 87. Note that the JCL specifies a DD that contains “simplexed” output cards (SWDSIMP) and a DD that contains “duplexed” output cards (SWDDUP). In this step, you **must** run MVCMAINT against the simplexed cards (SWDSIMP)!

   Also note that, in Step 2 on page 87, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in this step will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

   Figure 87 on page 162 shows example JCL to run SWSADMIN against the simplexed cards only if MVCs are selected.

5. **Run the Control-M/Tape CTIVIM batch job(s) to create the list of MVCs to be returned from offsite storage for reuse.**

6. **Using CTIVIM picklists, do the PTAM thing to take the MVCs ejected in Step 6 on page 82 to the offsite vault.**

7. **While you’re at the offsite vault...**

   ...pick up any MVCs which are to be moved back to the onsite library. This is a little hard to visualize, so think of it this way: We actually identified these MVCs in Step 2 on page 87 during **a previous run** of this procedure that occurred **greater than** nn **days ago**, where nn is the expiry period specified on the EXPIRYPERIOD statement. For more information, see “EXPIRYPERIOD” on page 167.

   Great, we’re done...that is, until the next time we get to cycle back into the defrag procedure described in “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 85.

Business Continuance

Okay, the Unthinkable has occurred, and the Production Site has taken a major hit. Now what? Well, the operations staff next wants to switch to business continuance mode. Note that because of the setup we did in “Normal Operations” on page 73, we can quickly and effectively resume operations because:

- All critical data is migrated and vaulted.
- A Recovery Site (which could be a vendor such as Comdisco) is standing by with VSM installed.

Because we did our homework on the setup, the switch to Business Continuance Mode at the Recovery Site is quick and straightforward via the MVC import shown in Figure 42 on page 90. After we switch to Business Continuance Mode, operations look like Figure 43 on page 90 and Figure 44 on page 91. We’ll use the following procedure to switch to Business Continuance Mode.

To switch to Business Continuance Mode at the Recovery Site, do the following:

1. **Create a new CDS at the Recovery Site.**

   This CDS reflects LIBGEN and VTCS configuration at the Recovery Site; we’ll populate this with VTV and MVC information in Step 4.

2. **Enter only critical MVCs into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

3. **Start an HSC audit on the panels we filled in Step 2.**

4. **Using the *most current* comprehensive Manifest File, run an import into the *new* CDS we created in Step 1.**

   We created the comprehensive Manifest File in Step 3 on page 81.

   Hint: When the import completes, we can start work using the critical data sets.

5. **Enter all remaining MVCs (and any standard Nearline volumes) into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

6. **Start an HSC audit on the panels we filled in Step 5.**

7. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Recovery Site VTSSs.**

8. **Now, and only now, start sending non-critical data to the Recovery Site VTSSs.**

   This completes this procedure, and we’re now up and running again. As soon as the Production Site is back in operation, run, do not walk, to “Business Resumption” on page 92.

Business Continuance - MVC Import

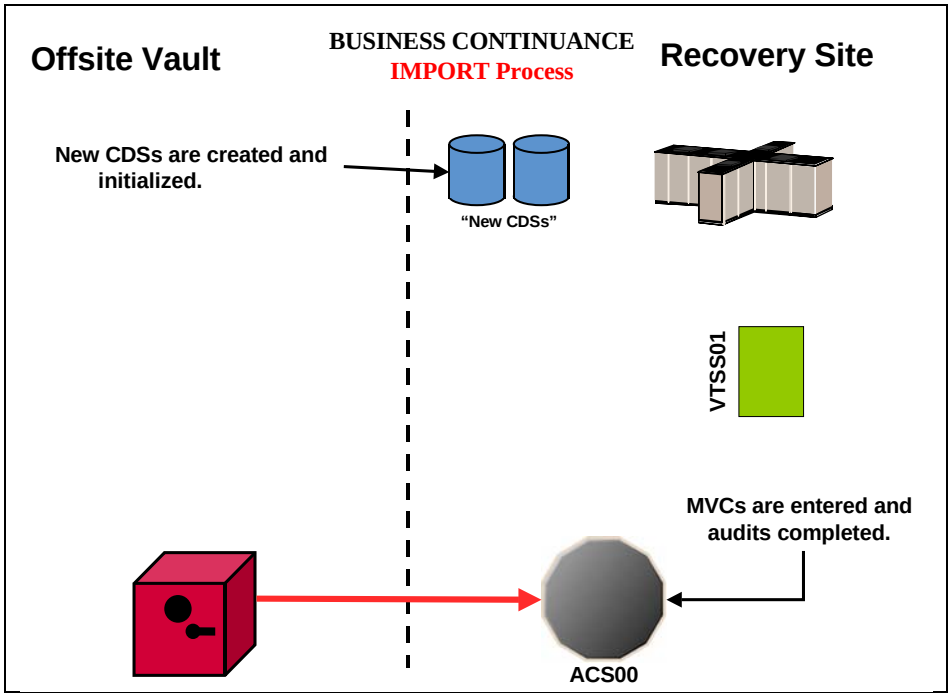


Figure 42. Business Continuance - MVC Import

Business Continuance - VTV Recall

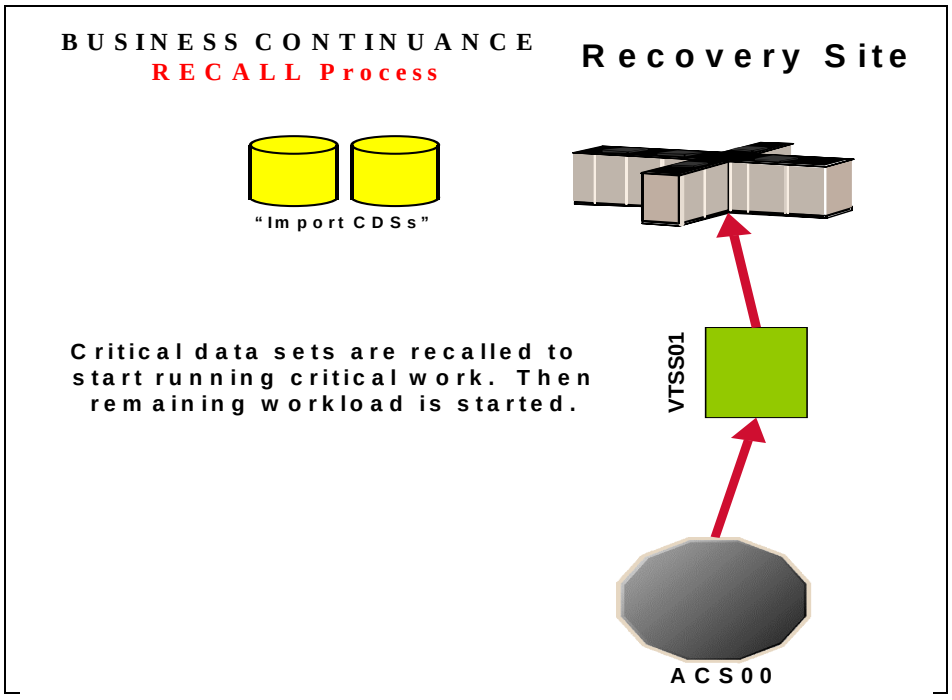


Figure 43. Business Continuance - VTV Recall

Business Continuance - VTV Migration

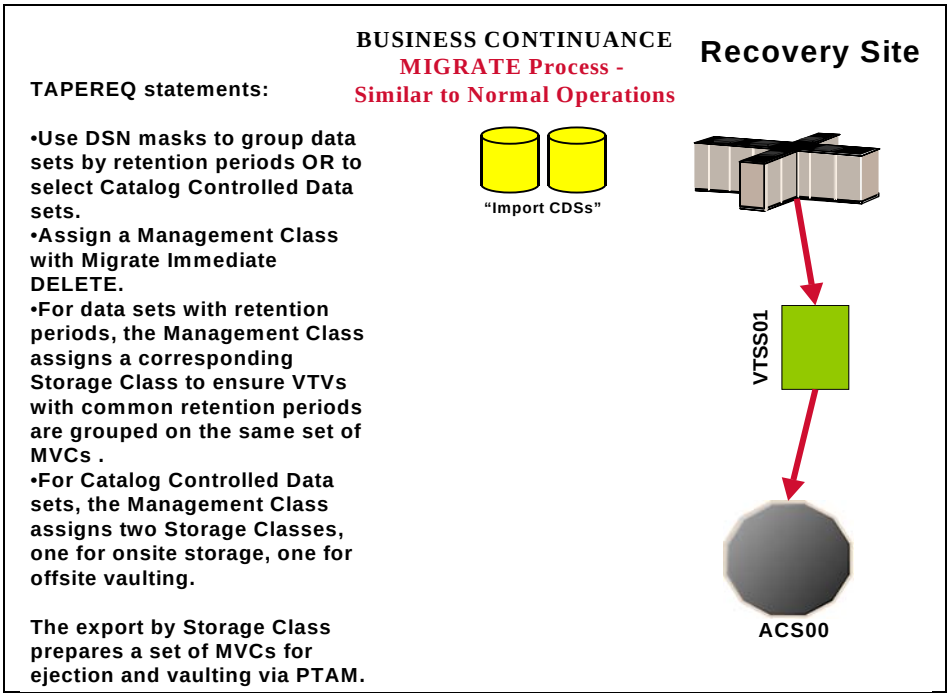


Figure 44. Business Continuance - VTV Migration

Business Resumption

The Production Site is back up as a data center, so it's time to get back to business as usual as shown in Figure 45 on page 93, Figure 46 on page 93, and Figure 47 on page 94.

The switch from Business Continuance to Business Resumption should look familiar, because it's very similar to what we did in "Business Continuance" on page 89...only this time we're using information and resources from the Recovery Site to recreate the Production Site VSM operation.

To resume normal operations:

1. **Clean the Production Site VTSSs.**

   This procedure, which is done by StorageTek hardware service, wipes out any extraneous VTVs that may be lingering around.

2. **Run an Export against all MVC ranges used at Recovery Site, using the Recovery Site CDS.**

Note: At this point, we're done with the Recovery Site...although we might want to keep it up and running until we reach Step 10.

3. **Create a new CDS at the Production Site.**
4. **Enter only critical MVCs into the Production Site ACS.**

   Fill complete rows, one panel at a time.

5. **Start an HSC audit on the panels we filled in Step 4.**
6. **Using the Manifest File from Step 2, run an import into the *new* CDS we created in Step 3.**

Hint: When the import completes, we can start work using the critical data sets.

7. **Enter all remaining MVCs (and any standard Nearline volumes) into the Production Site ACS.**

   Fill complete rows, one panel at a time.

8. **Start an HSC audit on the panels we filled in Step 7.**
9. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Production Site VTSSs.**

Note: After the VTVs that contain critical data sets are VTSS resident, you must enable **access** to the data sets on these VTVs, either by recataloging them, or using JCL statements such as `VOL=SER=vtvnnn`.

10. **Now, and only now, start sending new data to the Production Site VTSSs.**

Business Resumption - VTV Migration

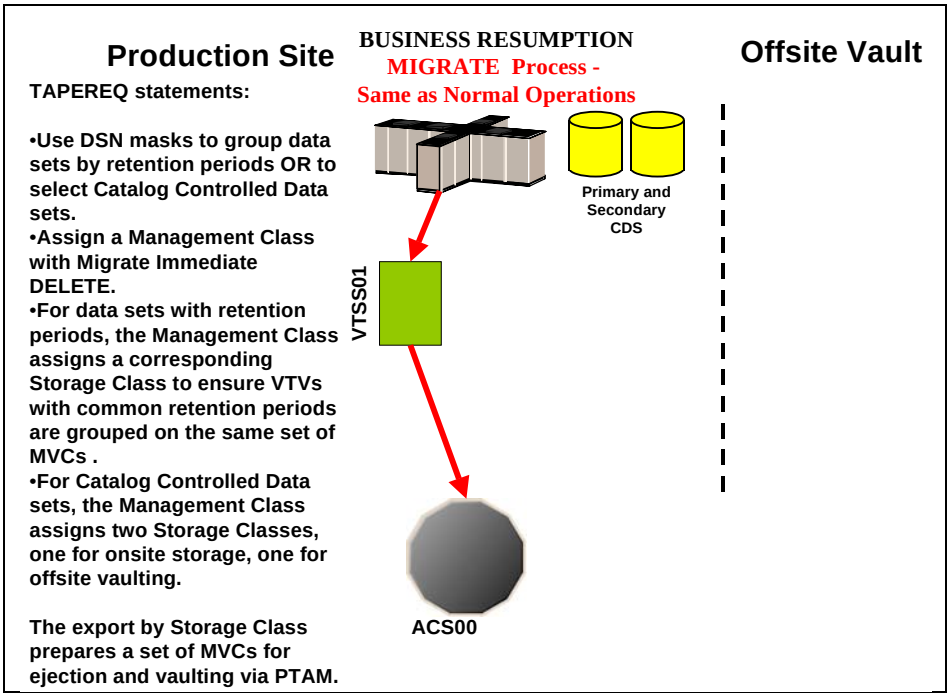


Figure 45. Business Resumption - VTV Migration

Business Resumption - VTV Recall

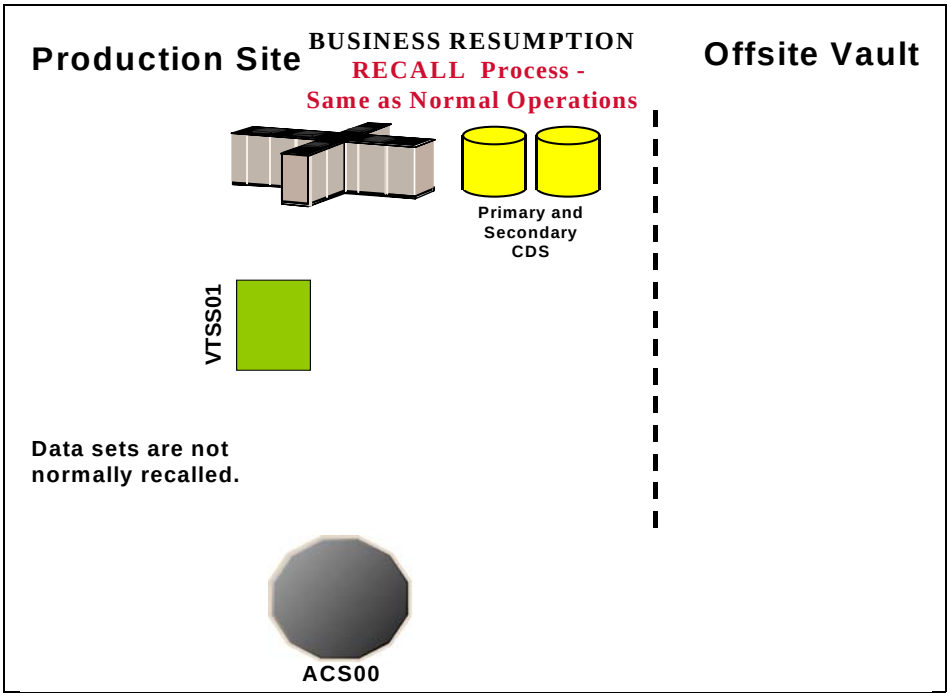


Figure 46. Business Resumption - VTV Recall

Business Resumption - MVC Export

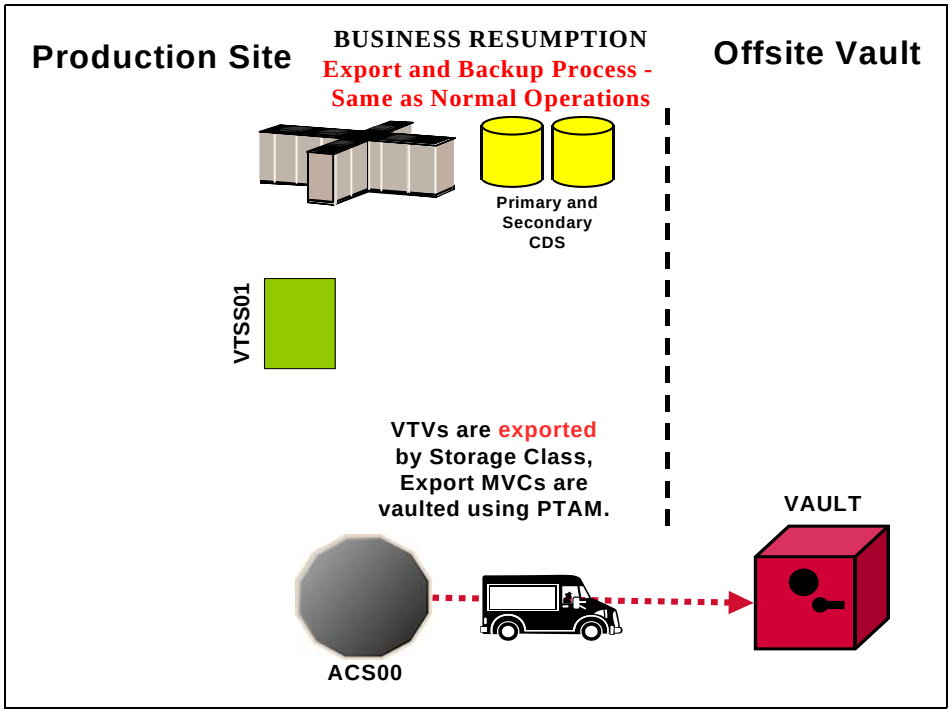


Figure 47. Business Resumption - MVC Export

Offsite Vaulting - The Pickup Truck Access Method for DFSMSrmm

This section tells how to use VSM and the PTAM ("Pickup Truck Access Method") to disaster-proof a single-site, offsite vault data processing operation so it can continue and resume operations after a catastrophic failure to the system. Throughout the procedures, you'll see how to do this for data sets with retention periods and for those without retention periods...specifically, they are catalog controlled data sets. The additional sub-flavor is that we're using a TMS to extract the date stamps, and the TMS is DFSMSrmm. The advantage of using a TMS is that there are accompanying StorageTek-written utilities that help automate the whole process.

As Figure 48 on page 97 shows, our example configuration consists of an MVS host at the Production Site, attached VTSS, single ACS, and an Offsite Vault. The vaulting of critical data works as follows:

- Data sets **with** retention periods are routed via `TAPEREQs` to VTVs by a series of Management Classes (keyed to retention periods) that specify `MIGRATE IMMEDIATE (DELETE)`. Each Management Class specifies a corresponding Storage Class so that the VTVs for that retention period are grouped on a common set of MVCs. After the migrations complete, you next run the `SWDIMS01` utility to create export MVCs to eject via ExLM for transport to the offsite vault. The VTVs on the export MVCs have expiry dates per the TMS (DFSMSrmm). As these VTVs expire, you use an ExLM Custom Volume Report and the `SWDIMS02` utility to identify MVCs with no current VTVs in the offsite vault. You then use DFSMSrmm `EDGHSKP` batch jobs to create a list of these MVCs to return them to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs with a current data generation.
- **Catalog controlled** critical data sets are routed via a `TAPEREQ` to VTVs by a single Management Class that specifies `MIGRATE IMMEDIATE (DELETE)`. This Management Class specifies a two corresponding Storage Classes, one for onsite storage, one for offsite vaulting. After the migrations complete, you next run the `SWDIMS01` utility against the offsite Storage Class to create export MVCs to eject via ExLM for transport to the offsite vault. When ExLM reports MVCs with space utilization of less than a user-specified value, you use the `SWDIMS02` utility to identify these MVCs. You then "logically drain" these MVCs (by draining the onsite MVC), use DFSMSrmm `EDGHSKP` batch jobs to create a pull list of the MVCs eligible for reuse, and then return the offsite MVCs to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs.

Caution: Please note the following gotcha. Some customers use HSC User Exit 06 to enter new tapes into the ACS and define them as scratch. This is exactly what we'd want for Nearline tapes, but scratch MVCs are a **not** a value add. So if you're entering MVCs and HSC User Exit 06 is set to define newly entered tapes as scratch, use the HSC `UEXIT` command to disable the exit before entering tapes to be used as MVCs.

"Offsite Vaulting - The Pickup Truck Access Method for DFSMSrmm" consists of the following subsections:

- "Normal Operations" on page 97, where we show what the Production Site looks like when the data is flowing smoothly. Here, we also tell how to set up the system to withstand a failure.
- "Business Continuance" on page 113, which is how to stay open for business if the Production Site takes a major hit.
- "Business Resumption" on page 116, which is how to get back to normal operations once the Production Site is up and running again.

Normal Operations uses the StorageTek VSM Vault Utilities (the `SWDIMS01` and `SWDIMS02` utilities), so before diving into Normal Operations, let's turn the page and install these utilities.

Normal Operations

On page 95, we gave an overview of the system, disaster recovery needs, and operations to meet those needs. Figure 48 and Figure 49 on page 98 show the system during normal operations for VTV migration and recall. Figure 50 on page 98 shows the export and physical transport to the offsite vault of the MVCs that contain the critical data. There is, as we said, Some Assembly Required, so let's turn to "Migration, Export, Offsite Vault, and MVC Reuse" on page 99 for the details.

Normal Operations - VTV Migration

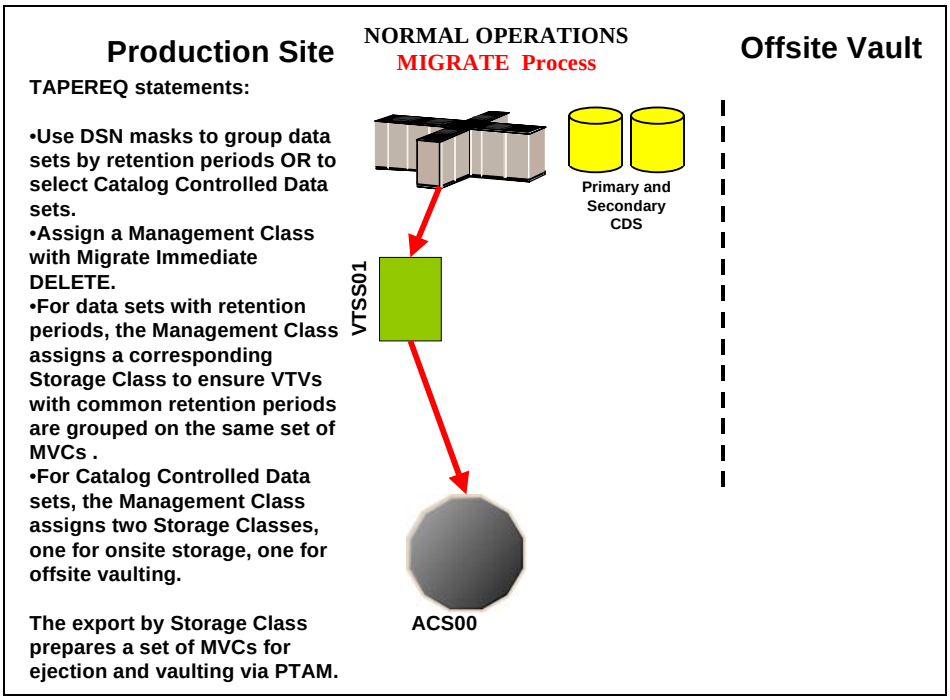


Figure 48. Normal Operations - VTV Migration

Normal Operations - VTV Recall

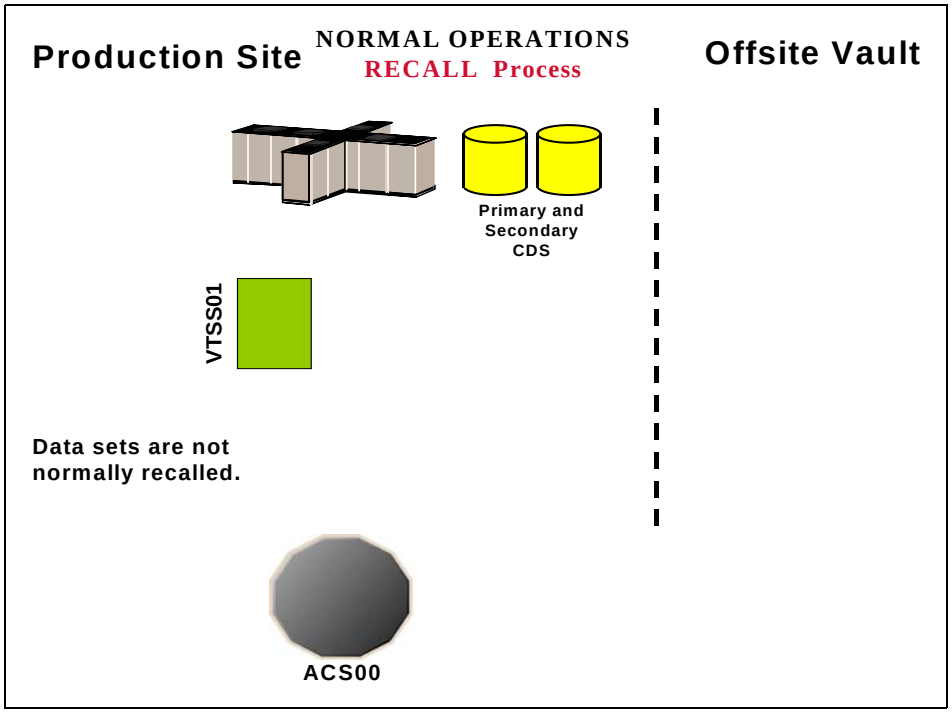


Figure 49. Normal Operations - VTV Recall

Normal Operations - MVC Export

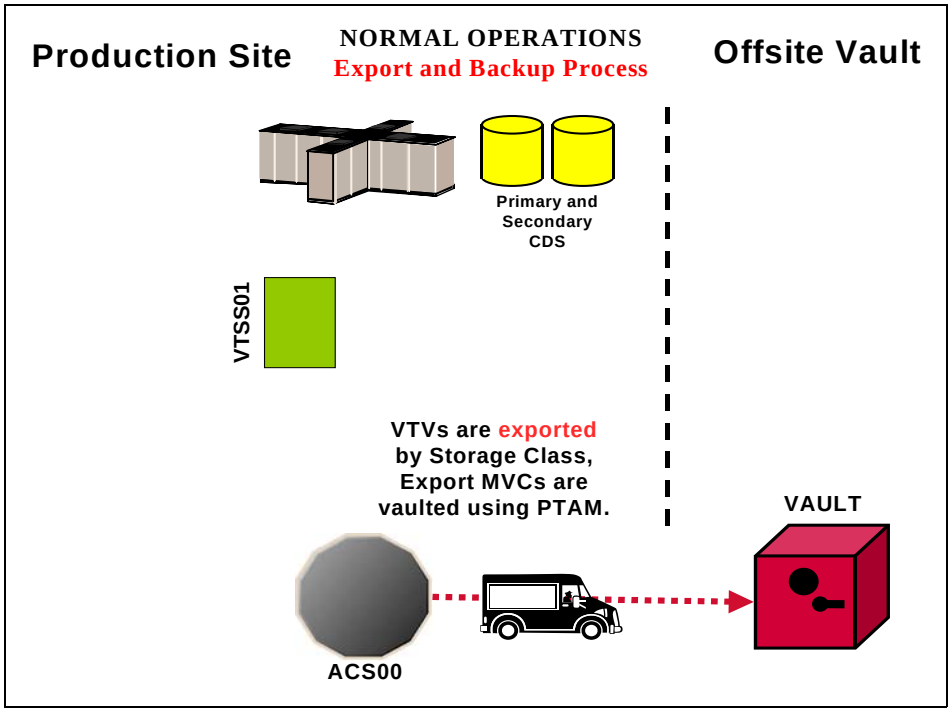


Figure 50. Normal Operations - MVC Export

Migration, Export, Offsite Vault, and MVC Reuse

In “Normal Operations” on page 97, we showed the system under normal operations. This section tells how to route critical data **in data sets with retention periods** and **catalog controlled data sets** to VTVs, migrate them to a common set of MVCs, and vault the MVCs offsite.

Table 17 on page 100 and Table 18 on page 100 are based on the following crucial differences in that data sets that you want to vault:

- First, in Table 17 on page 100, for **catalog controlled data sets**, we’re again showing the use of a data set name mask on TAPEREQ. As above, use other data set characteristics that you specify on TAPEREQ if that works better. Note that with catalog controlled data sets, the TAPEREQ specifies a single Management Class that duplexes the data via two Storage Classes. This produces an offsite MVC and an onsite MVC with the critical data, which, as we’ll see later, is critical to reusing and recycling MVCs.
- In Table 18 on page 100, for **data sets with retention periods**, we’re showing a situation where you have multiple data set name masks, each of which corresponds to a different retention period for the data sets selected by that mask. The fewer of these, the better (read what the VTCS documentation says about Too Many Storage Classes). That is, for *each* data set name mask you need a corresponding Management Class and Storage Class.

  With TAPEREQ, in fact, you have lots of choices...for example, you can key on jobname or stepname, if that works better. If you want to key on retention periods, data set name is probably the right move, however. We just want to make sure you know that you **do** have the flexibility to key on different data set groupings with different characteristics.

So at this point your question is “What do I do if I hav***e both*** flavors of data sets?” Well, we thought of that, too, because in the real world this is a very likely situation. Back in the MVC reuse section, which is one of the key areas to get right, we break that part of the process down for you. You can read about all of that in “Returning MVCs from the Offsite Vault for Reuse” on page 107. Another useful tool, however, especially for all you right-brainers, is the overall process flow for “mixed” shops that we show in “Offsite Vaulting Configuration Record” on page 177.

- In this example, we're selecting the MVCs for offsite vaulting from a Named MVC Pool. This is optional, but it's not a bad idea because:
  - All the non-critical jobs select MVCs either from another Named MVC Pool or from the overall MVC Pool (except for the Named Pool used for offsite vaulting).
  - At the Recovery Site, you only have to define the MVCs in the Named Pool.

  Once again, however, read the *caveats* in the VTCS documentation about the potential gotchas with Named MVC Pools.
- Finally, throughout the procedures, we refer you back to Table 18 or Table 17. In fact, what we *really* mean is your versions of these tables, which you'll have to create to match your shop's situation...as recorded in "Offsite Vaulting Configuration Record".

Table 17. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Catalog Controlled Data Sets without Retention Periods

Management Class	**Storage Classes**	**Use**	**Named MVC Pool**	**TAPEREQ Data Set Mask**
mgmtcat	storon storoff	On site MVCs Offsite MVCs	poolcat	maskcat

Table 18. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Data Sets with Retention Periods

Management Class (mgmt-class-name)	**Retention Period RETPD(LT,n)**	**Storage Class** (stor-clas-name)	**Named MVC Pool** (poolname)	**TAPEREQ Data Set Mask** (mask)
mgmtret1	1 to 7 days	storret1	*poolret1*	maskret1
		. . .		
mgmtretn	Greater than one century	storretn	*poolretn*	maskretn

- Let's not forget the SWDTMS01 and SWDTMS02 utilities, which we'll use in the following sections. For some reference information, see "SWDTMS01 Utility" on page 144 and "SWDTMS02 Utility" on page 155.

  The whole of Normal Operations takes some doing, so let's break it down into the following sections:

  - "Setting Up the System" on page 101
  - "Creating the Export MVCs and Vaulting Them Offsite" on page 105
  - "Returning MVCs from the Offsite Vault for Reuse" on page 107

Setting Up the System

To set up the system:

1. **First, we'll enable the Advanced Management Feature.**

   The Advanced Management Feature is required to implement Storage Classes, run the EXPORT and IMPORT utilities, and use selected Management Class parameters (such as MIGPOL).

2. **Let's build the Management Classes that assign a corresponding Storage Class and do an immediate migrate with delete.**

   - **For data sets with retention periods**, *each* of the MGMTclas statements (you create a separate statement for each retention period) looks something like Figure 51:

```
MGMT NAME(mgmtretn) IMMED(DELETE) MIGPOL(storretn) DELSCR(YES)
```

Figure 51. Management Classes for VTVs with Critical Data

   - **For catalog controlled data sets**, the MGMTclas statement looks something like Figure 52:

```
MGMT NAME(mgmtcat) IMMED(DELETE) MIGPOL(storon,storoff) DELSCR(YES)
```

Figure 52. Management Classes for VTVs with Critical Data - Catalog Controlled Data Sets

   In Figure 51 and Figure 52, each Management Class specifies immediate migrate delete, which is designed to immediately put VTVs in this Management Class on the migration queue and delete the VTSS copy once it is migrated. This ensures quick migration and frees the VTSS buffer. We also assign the corresponding Storage Class to the MVCs that contain the migrated VTVs as shown in Table 18. on page 100 or Table 17 on page 100.

Warning: Note that this Management Class specifies DELSCR(YES). If you haven't already done so, **go back and read** the planning information in "DELSCR Considerations" on page 6.

3. **Next, we'll create the Storage Classes that own the MVCs that contain the migrated VTVs.**

   - **For data sets with retention periods**, each of the STORclas statements (you create a separate statement for each retention period) looks something like Figure 53:

```
STOR NAME(storretn) MVCP(poolret)
```

Figure 53. Storage Classes for Data Sets with Retention Periods

   For the Storage Class names to plug in, see Table 18. on page 100. In Figure 53, the STORclas statement associates each Storage Class with the specified Named MVC Pool.

   - **For catalog controlled data sets**, the STORclas statements looks something like Figure 54:

```
STOR NAME(storon) MVCP(poolcat)
STOR NAME(storoff) MVCP(poolcat)
```

Figure 54. Storage Classes for Catalog Controlled Data Sets

   For the Storage Class names to plug in, see Table 17. on page 100. In Figure 54, the STORclas statements associate each Storage Class with the specified Named MVC Pool.

4. **Next, we'll use the MGMTDEF command to load the MGMTclas and STORclas control statements we created in Step 2 on page 101 and Step 3 on page page 102.**

5. **Now let's create TAPEREQ statements to route the critical data to VSM and assign the corresponding Management Class to the data.**

   - **For data sets with retention periods**, each of the TAPEREQ statements (you create a separate statement for each retention period) looks something like Figure 55:

```
TAPEREQ DSN(maskretn) MEDIA(VIRTUAL) MGMT(mgmtretn) RETPD(LT,n)
```

Figure 55. TAPEREQ Statements for Data Sets with Retention Periods

Note: The RETPD parameter is not supported under JES3, so use another TAPEREQ data set selection method (for example, JOBNAME).

   See Table 18. on page 100 for the correct values to fill in for Figure 55.

   - **For catalog controlled data sets**, the TAPEREQ statement looks something like Figure 56:

```
TAPEREQ DSN(maskcat) MEDIA(VIRTUAL) MGMT(mgmtcat)
```

Figure 56. TAPEREQ Statement for Catalog Controlled Data Sets

   See Table 17. on page 100 for the correct values to fill in for Figure 56.

6. **Use the TREQDEF command to load the TAPEREQ control statements we created in Step 5.**

7. **If they aren't already in place, set up the DFSMSrmm vault codes for the exported MVCs.**

   DFSMSrmm uses the data set name (and optionally the name of the job that creates the data set) to assign vaulting codes. For more information on setting up vaulting codes to achieve the desired vaulting assignments and rotation, see *DFSMSrmm User's Guide*.

   The name of the data set that the SWDIMS01 utility assigns to the MVCs in the DFSMSrmm control data set is:

   DSN=prefix.storclas.MVCTAPE.RESERVED

   Where:

   - prefix is the HLQ specified in the MvcDsnPreFiX parameter of the STORclas statement. For more information, see "EXPiryPeRioD Statement Example" on page 168.
   - storclas is a Storage Class created in Step 3 on page 102.

   For example, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and LE8DAYS for the Storage Class for retention periods of less than 8 days, the SWDIMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.LE8DAYS.MVCTAPE.RESERVED

   Similarly, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and CATOFF for the Storage Class for "offsite" MVCs with catalog controlled data sets, the SWDIMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.CATOFF.MVCTAPE.RESERVED

8. **There is one last item...**

   ...which is defining the MVCs to the TMS. As it says in the VTCS documentation:

   > "Access to the MVCs via an RTD bypasses the MVS intercepts put in place by the tape management system so that it does *not* record within its database any access to the MVCs by VSM and does *not* automatically provide protection against inadvertent overwrites of non-expired data on MVCs. Therefore, if you choose to define MVCs to the tape management system, StorageTek **strongly recommends** that you define them as non-scratch, non-expiring volumes."

   Well, that isn't going to work so well for MVCs that you want to reuse, so we have to find another way. So if you want to define the MVCs to DFSMSrmm and adequately protect them, try this:

   a. Do a format extend to add the MVC volsers to the DFSMSrmm control data set.

   b. Second, define the MVCs as a scratch subpool to DFSMSrmm.

   c. Finally, write a rule that says that *only* HSC/VTCS can write to the MVCs as scratches.

   Section 1, Setting Up the System, is done, so let's proceed to "Creating the Export MVCs and Vaulting Them Offsite" on page 105.

To create the export MVCs and vault them offsite:

1. **Ensure that all VTVs with critical data are migrated.**

   The TAPEREQ statements in Step 5 on page 102 are the triggers to start the VTV migrations to the MVCs we'll export. But how do we know when all the VTVs with critical data are migrated? Basically, we write a "watchdog" program that monitors VTV migration and reports when the *last* critical VTV is migrated.

2. **Run the SWDIMS01 utility.**

   The SWDIMS01 utility, working together with the JCL you create to invoke it, does the following:

   - **Runs the EXPORT utility** against **all** Storage Classes we created in Step 3 on page 102 **and creates a comprehensive Manifest File** that includes all these Storage Classes. We use the STORclas statement of the parameter file to specify one or more Storage Classes. For more information, see "EXPiryPeRioD Statement Example" on page 168.

   Note: If an MVC in a specified Storage Class is in use when the SWDIMS01 utility runs, it will be skipped and will not be exported until the next time the utility runs.

   The comprehensive Manifest File is used, if needed, in Business Continuance mode (see Step 4 on page 113)...so it's probably a good idea to have one copy of the file on disk and another on tape stored offsite.

   - **Creates an MVC update file**, which is used as input to the batch TSO/E job in Step 3 on page 106, to set data set names and expiration dates for the MVCs to be sent offsite. These commands set the data set expiration date to PERMANENT.
   - **Creates a VTV update file**, which can optionally be used as input to the batch TSO/E job in Step 3 on page 106, to change the TMS VTV records.

     These control cards reflect the state of the VTVs being sent offsite, and it's probably not a good idea to alter them. If, for some reason, you have a need to change the state of VTVs being sent offsite, contact StorageTek Software Support for information on the use of these cards.

Hint: Good Practices suggest that we **only** want to vault volumes that are resident in an ACS. To do this, run the SWDIMS01 utility with the STORCLAS ALL(OFF) parameter of the parameter file. If, however, you want to see a selection list of all volumes that **should** be vaulted, whether they are ACS resident, use the STORCLAS ALL(ON) parameter. For more information, see "EXPiryPeRioD Statement Example" on page 168. **Note that**, even if you specify the ALL(OFF) parameter, the Manifest File is **always** "comprehensive."

For example JCL for the SWDIMS01 utility, see Figure 78 on page 146 and Figure 79 on page 147.

3. **Next, we run the batch TSO/E program to update the DFSMSrmm control data set with MVC information.**

   The batch TSO/E program uses the input from Step 2 on page 105 to update the MVC volume records in the DFSMSrmm control data set with a dummy data set name, expiration date, creation date, and so forth.

Hint: In Step 2, if the SWDIMS01 utility **does not** select any MVCs for update, SWDIMS01 ends with a return code of 4. Because the DFSMSrmm control data set update in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS01 runs successfully, selects MVCs, and completes with a return code of 0 as shown in Figure 80 on page 148.

4. **To assign the vault codes and rotation to the MVCs, run the DFSMSrmm EDGHSKP batch jobs.**

   ...where we set up the vault codes in Step 7 on page 103 and updated the DFSMSrmm control data set with MVC information in Step 3 on page 106.

   For more information on the DFSMSrmm EDGHSKP, see *DFSMSrmm Administrator and Operator Guide*.

5. **Keying on the Vault Code from Step 7 on page 103, we use ExLM to eject the MVCs from the ACS.**

   Figure 81 on page 149 shows example JCL to use ExLM to eject the MVCs.

6. **Finally, we use PTAM (Pickup Truck Access Method) to do the offsite vaulting of the MVCs we ejected in Step 5.**

   Because we have a finite number of MVCs, when VTVs go non-current on exported MVCs, we need a way to cycle these MVCs back into the system for reuse. For that procedure, see "Returning MVCs from the Offsite Vault for Reuse" on page 107.

Returning MVCs from the Offsite Vault for Reuse

In “Creating the Export MVCs and Vaulting Them Offsite” on page 105, we migrated VTVs with critical data to MVCs and exported those MVCs to the Offsite Vault. We don’t have an infinite number of MVCs for vaulting, so we need a way to recycle vaulted MVCs back into the system when some or all VTVs on these MVCs go non-current. And there are three flavors, so pick the one that works for your shop and follow the referenced procedure:

1. **Flavor 1 - Sites with *only* data sets with retention periods**, when ExLM reports that all VTVs on an MVC have reached their expiry dates in DFSMSrmm, the current VTV count goes to zero and we can return the MVC to the Production Site for reuse. For procedures, see “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 111.

2. **Flavor 2 - For sites with *only* catalog controlled data sets**, we use an ExLM report to look for vaulted MVCs that have a usage percentage less than a percentage we specify. In other words, ExLM looks for MVCs that have enough free space to make it worthwhile to reuse these MVCs. What’s a reasonable percentage to specify? The answer, of course, is “it depends on the needs of your shop.” Specify a high percentage and you’ll free up lots of MVCs...at the cost of high VSM/system activity. Specify a low percentage and you’ll have lots of fragmented MVCs in the offsite vault. Try starting with 25% and adjust as needed.

   The way we reuse these MVCs is...well, it’s complex, so let’s slow down and see how it works:

   1. MVCs with less than `n` percent usage are marked `LOST`. Now why would I do that? Because I have an equivalent on site copy and I don’t want to put an operator through mounting the offsite MVCs to drain it, see number 2.

   2. I try to run a drain against the offsite MVC, it’s lost, so instead VTCS “logically drains” the offsite MVC by recalling all its VTVs from the MVC copy that is onsite. Because the recalled VTVs have the DR Management Class, they are migrated to a new MVC that can be taken offsite. The new MVC is now the input to Step 2 on page 105, which detects MVCs with Storage Classes that require offsite vaulting.

      So VTCS “logically drained” the offsite MVC, which is goodness because the offsite MVC is now 0% full. It is now eligible for selection by Step 1 on page 111 and is returned per Step 7 on page 112.

   And for the step-by-step implementation of this process, see “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 109.

Note that you are not done until, as it says in Step 4 on page 110, you next complete the procedure described in “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 111...because the logical draining has created MVCs with a zero current VTV count. Is that cool, or what?

Hint: How often do you want to do this defragmentation? The answer is “it depends on the needs of your shop and your level of vaulting/reuse activity”...but doing this once a week is probably a good starting point.

3. **Flavor 3 - For "mixed" sites with *both* catalog controlled data sets *and* data sets with retention periods,** you do the same thing (and for the same reasons) that you do for sites with only catalog controlled data sets. That is, you first do the procedure in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 109, followed by the procedure in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 111.

Note: Each of the three procedures described above **requires** the standard `EDGHSKP` update process described in *DFSMSrmm Administrator and Operator Guide*.

Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets.

To Recycle Offsite MVCs That Contain VTVs With Catalog Controlled Data Sets:

1. **First, let's run an ExLM Custom Volume Report...**

   ...to find MVCs with less than a specified percent used.

   Figure 82 on page 157 shows a JCL example that generates an ExLM report that selects vaulted MVCs with less than a specified percent used. In Figure 82 on page 157:

   - `DSN=prefix.storoff.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the DFSMSrmm control data set, as described in Step 7 on page 103.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "EXPiryPeRioD Statement Example" on page 168.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specify in Step 1 on page 111. The difference is in the *storclas* value for the middle qualifier on the DSN. In this step, it is the single DSN for the data set assigned to all MVCs that contain VTVs with catalog controlled data sets. In this step, you want to drain ***only*** the MVCs with catalog controlled data sets, not the MVCs with data sets with retention periods. If you go all the way back to Step 3 on page 102, you'll see that it is, in fact, `storoff`, which is the Storage Class for the offsite (vaulted) MVCs.

   - `loccode` is the location code for the offsite vault.
   - `percent` is the specified percentage use. See page 107 for a discussion of what constitutes a reasonable value.
   - `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 84 on page 159 shows example JCL where the `SWDIMS02` utility:

   - Creates the appropriate `MVCMAINT` commands to mark as `LOST` the MVCs selected in Step 1.
   - Creates the appropriate `MVCDRAIN` commands to make all the swell stuff happen that we described starting in number 1. on page 107.

3. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 109 to enable draining the fragmented offsite MVCs.**

Caution: Here we need to take a look at Figure 85 on page 160. This is the JCL to run the SWDIMS02 utility in Step 2 on page 109. Note that the JCL specifies a DD that contains "simplexed" output cards (SWDSIMP) and a DD that contains "duplexed" output cards (SWDDUP). In this step, you **must** run MVCMAINT against the duplexed cards (SWDDUP)!

Also note that, in Step 2, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

Figure 85 on page 160 shows example JCL to run SWSADMIN against the duplexed cards only if MVCs are selected.

Finally, **Also note that**...we're running a drain to recall the current VTVs from the MVC...remember that these are MVCs that are x percent used. How does this actually happen? For the long version, see page 107. Here's another data point:

> After Step 3 completes, the still current VTV, which had one copy on the offsite MVC which was drained, will have one copy on the onsite MVC, and a new copy on another onsite MVC which has the same Storage Class as the original offsite MVC. In the meantime, the offsite MVC will have had its storage class removed. This new onsite DR MVC will be picked up in the next SWDIMS01 run, and will be sent offsite.

And, additionally:

> The drain process also clears the LOST status of the offsite MVC.

4. **Don't stop now...**

...because, for all the good reasons we gave you back on page 107, this process **feeds into** and **requires** that you next complete the procedure described in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 111.

Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods.

To Recycle Offsite MVCs That Contain VTVs With Data Sets with Retention Periods:

1. **First, let's run an ExLM Custom Volume Report to find any MVCs vaulted for data sets with retention periods with no current VTVs....**

   Figure 83 on page 158 shows a JCL example that generates an ExLM report that selects vaulted MVCs with no current VTVs. In Figure 83 on page 158:

   - `DSN=prefix.storclas.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the DFSMSrmm control data set, as described in Step 7 on page 103.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "EXPiryPeRioD Statement Example" on page 168.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specified in Step 1 on page 109. The difference is in the *storclas* value for the Second Level Qualifier on the DSN. In this step, we want to detect all offsite MVCs with 0 VTVs. Therefore, the DSN mask in the ExLM job should be "`prefix.**`".

   - `loccode` is the location code for the offsite vault.

   `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 86 on page 161 shows example JCL where the `SWDIMS02` utility:

   - Creates the DFSMSrmm TSO/E commands to modify DFSMSrmm volume records for MVCs. These commands set the Expiration Date specified on the `EXPIRYPERIOD` statement to allow a return of the MVC from the vault. For more information, see "EXPIRYPERIOD" on page 167.
   - Creates the appropriate `MVCMAINT` commands to reset the Readonly bit to make these MVCs available for reuse after they're reentered into the ACS.
   - Creates the appropriate `MVCDRAIN` commands to...trust us, it's a good idea, and we'll explain it all in Step 4 on page 112.

3. **Run the batch TSO/E program to process the commands generated in Step 2...**

   ...to update the DFSMSrmm control data set accordingly. We do this, together with some other fun `SWSADMIN` stuff, as shown in the example in Figure 87 on page 162.

4. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 111 to reset the readonly status of the MVCs that will be returning to the production site...**

   ...and to also do a logical drain on those MVCs. “Now why,” you ask yourself, “Is he telling me to drain an *empty* MVC?” And the answer is that invoking MVCDRAIN does more than just draining VTVs, it also affects an MVC state change, which in this case is a Good Thing. Specifically, the MVDRAIN operation removes the Storage Class from the MVC, which prevents the MVC from being selected when SWDIMS01 is exporting MVCs...and also makes the MVC available to the universe, not just that Storage Class. Don’t worry, these empty MVCs will not have to be mounted to be logically drained.

   Caution: Here we need to take a look at Figure 86 on page 161. This is the JCL to run the SWDIMS02 utility in Step 2 on page 111. Note that the JCL specifies a DD that contains “simplexed” output cards (SWDSIMP) and a DD that contains “duplexed” output cards (SWDDUP). In this step, you **must** run MVCMAINT against the simplexed cards (SWDSIMP)!

   Also note that, in Step 2 on page 111, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in this step will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

   Figure 87 on page 162 shows example JCL to run SWSADMIN against the simplexed cards only if MVCs are selected.

5. **Run the DFSMSrmm EDGHSKP batch job(s) to create the list of MVCs to be returned from offsite storage for reuse.**

6. **Using EDGHSKP picklists, do the PTAM thing to take the MVCs ejected in Step 5 on page 106 to the offsite vault.**

7. **While you’re at the offsite vault...**

   ...pick up any MVCs which are to be moved back to the onsite library. This is a little hard to visualize, so think of it this way: We actually identified these MVCs in Step 2 on page 111 during **a previous run** of this procedure that occurred **greater than** nn **days ago**, where nn is the expiry period specified on the EXPIRYPERIOD statement. For more information, see “EXPIRYPERIOD” on page 167.

   Great, we’re done...that is, until the next time we get to cycle back into the defrag procedure described in “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 109.

Business Continuance

Okay, the Unthinkable has occurred, and the Production Site has taken a major hit. Now what? Well, the operations staff next wants to switch to business continuance mode. Note that because of the setup we did in “Normal Operations” on page 97, we can quickly and effectively resume operations because:

- All critical data is migrated and vaulted.
- A Recovery Site (which could be a vendor such as Comdisco) is standing by with VSM installed.

Because we did our homework on the setup, the switch to Business Continuance Mode at the Recovery Site is quick and straightforward via the MVC import shown in Figure 57 on page 114. After we switch to Business Continuance Mode, operations look like Figure 58 on page 114 and Figure 59 on page 115. We’ll use the following procedure to switch to Business Continuance Mode.

To switch to Business Continuance Mode at the Recovery Site, do the following:

1. **Create a new CDS at the Recovery Site.**

   This CDS reflects LIBGEN and VTCS configuration at the Recovery Site; we’ll populate this with VTV and MVC information in Step 4.

2. **Enter only critical MVCs into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

3. **Start an HSC audit on the panels we filled in Step 2.**

4. **Using the *most current* comprehensive Manifest File, run an import into the *new* CDS we created in Step 1.**

   We created the comprehensive Manifest File in Step 2 on page 105.

   Hint: When the import completes, we can start work using the critical data sets.

5. **Enter all remaining MVCs (and any standard Nearline volumes) into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

6. **Start an HSC audit on the panels we filled in Step 5.**

7. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Recovery Site VTSSs.**

8. **Now, and only now, start sending non-critical data to the Recovery Site VTSSs.**

   This completes this procedure, and we’re now up and running again. As soon as the Production Site is back in operation, run, do not walk, to “Business Resumption” on page 116.

Business Continuance - MVC Import

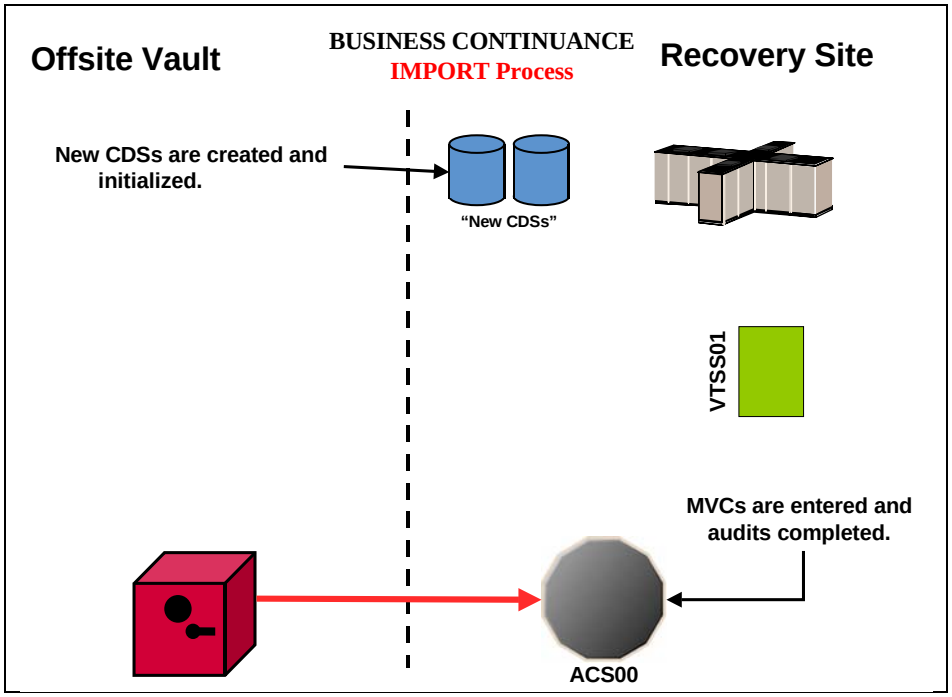


Figure 57. Business Continuance - MVC Import

Business Continuance - VTV Recall

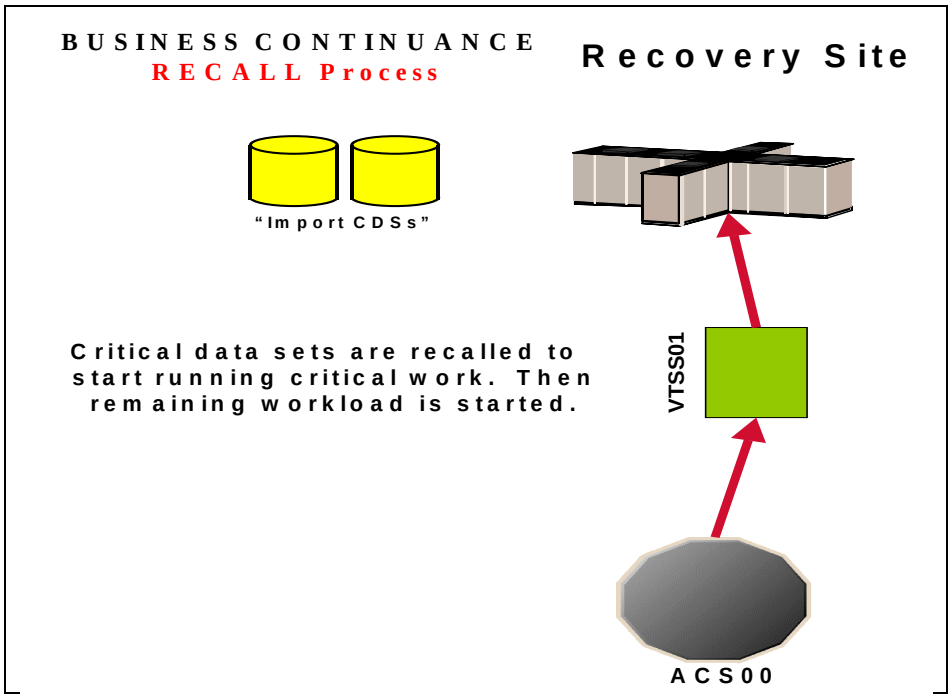


Figure 58. Business Continuance - VTV Recall

Business Continuance - VTV Migration

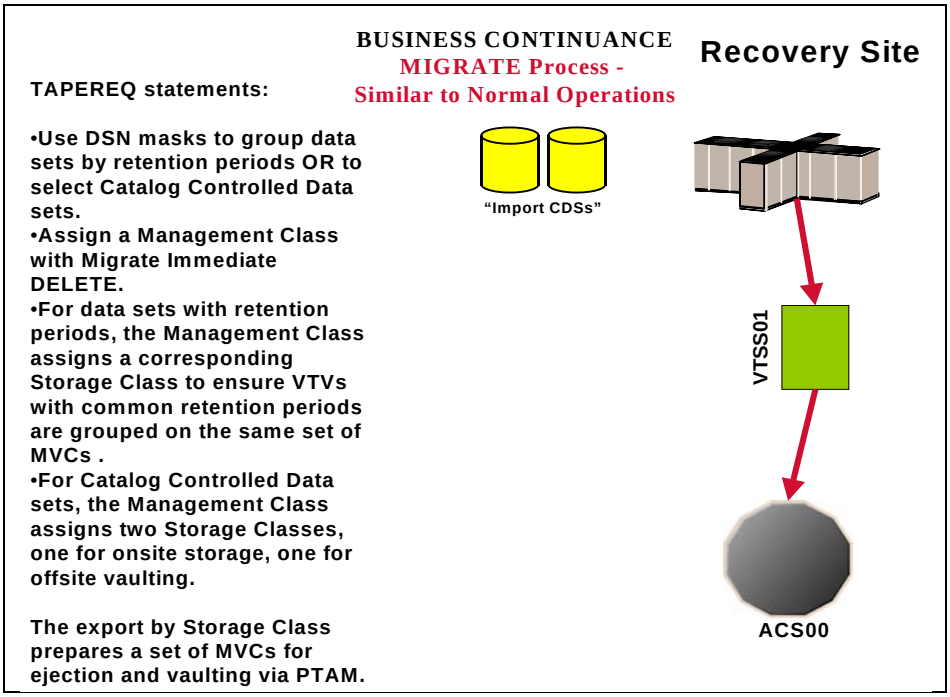


Figure 59. Business Continuance - VTV Migration

Business Resumption

The Production Site is back up as a data center, so it's time to get back to business as usual as shown in Figure 60 on page 117, Figure 61 on page 117, and Figure 62 on page 118.

The switch from Business Continuance to Business Resumption should look familiar, because it's very similar to what we did in "Business Continuance" on page 113...only this time we're using information and resources from the Recovery Site to recreate the Production Site VSM operation.

To resume normal operations:

1. **Clean the Production Site VTSSs.**

   This procedure, which is done by StorageTek hardware service, wipes out any extraneous VTVs that may be lingering around.

2. **Run an Export against all MVC ranges used at Recovery Site, using the Recovery Site CDS.**

Note: At this point, we're done with the Recovery Site...although we might want to keep it up and running until we reach Step 10.

3. **Create a new CDS at the Production Site.**
4. **Enter only critical MVCs into the Production Site ACS.**

   Fill complete rows, one panel at a time.

5. **Start an HSC audit on the panels we filled in Step 4.**
6. **Using the Manifest File from Step 2, run an import into the *new* CDS we created in Step 3.**

Hint: When the import completes, we can start work using the critical data sets.

7. **Enter all remaining MVCs (and any standard Nearline volumes) into the Production Site ACS.**

   Fill complete rows, one panel at a time.

8. **Start an HSC audit on the panels we filled in Step 7.**
9. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Production Site VTSSs.**

Note: After the VTVs that contain critical data sets are VTSS resident, you must enable **access** to the data sets on these VTVs, either by recataloging them, or using JCL statements such as `VOL=SER=vtvnnn`.

10. **Now, and only now, start sending new data to the Production Site VTSSs.**

Business Resumption - VTV Migration

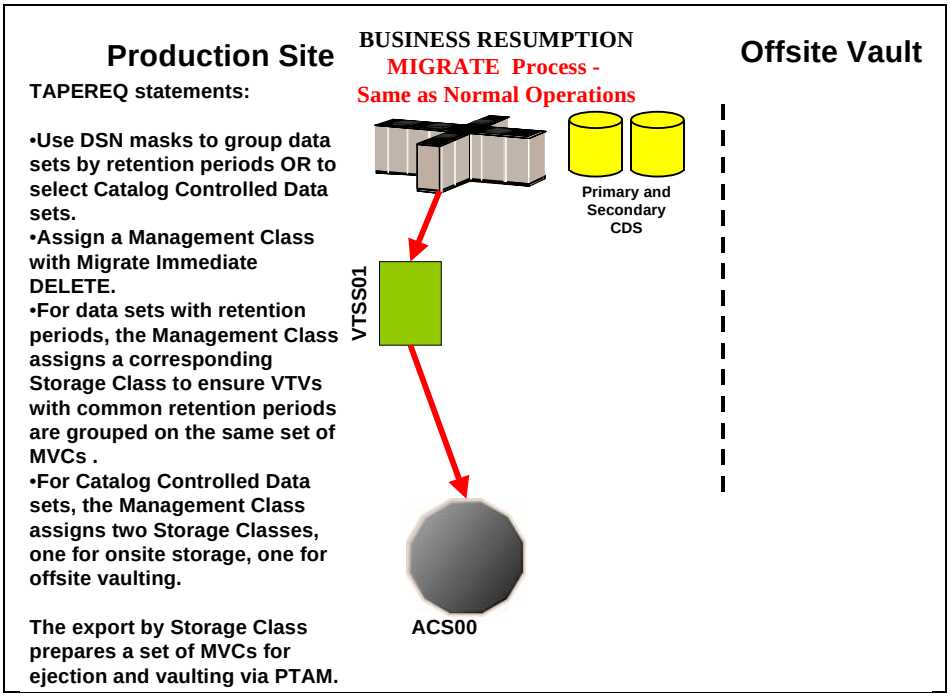


Figure 60. Business Resumption - VTV Migration

Business Resumption - VTV Recall

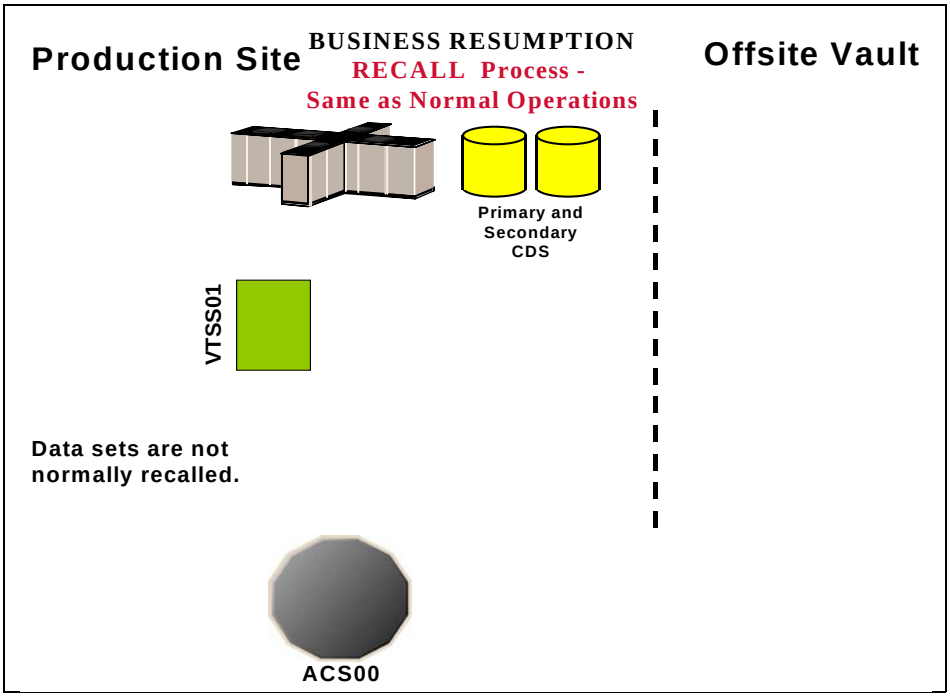


Figure 61. Business Resumption - VTV Recall

Business Resumption - MVC Export

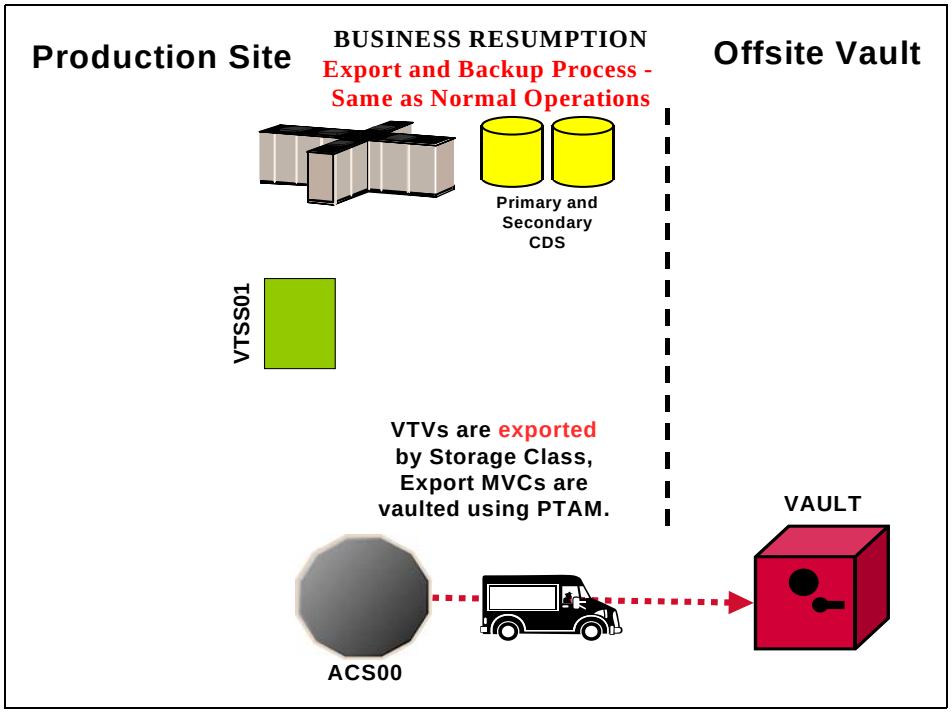


Figure 62. Business Resumption - MVC Export

Offsite Vaulting - The Pickup Truck Access Method for Zara

This section tells how to use VSM and the PTAM ("Pickup Truck Access Method") to disaster-proof a single-site, offsite vault data processing operation so it can continue and resume operations after a catastrophic failure to the system. Throughout the procedures, you'll see how to do this for data sets with retention periods and for those without retention periods...specifically, they are catalog controlled data sets. The additional sub-flavor is that we're using a TMS to extract the date stamps, and the TMS is Zara. The advantage of using a TMS is that there are accompanying StorageTek-written utilities that help automate the whole process.

As Figure 63 on page 121 shows, our example configuration consists of an MVS host at the Production Site, attached VTSS, single ACS, and an Offsite Vault. The vaulting of critical data works as follows:

- Data sets **with** retention periods are routed via `TAPEREQs` to VTVs by a series of Management Classes (keyed to retention periods) that specify `MIGRATE IMMEDIATE (DELETE)`. Each Management Class specifies a corresponding Storage Class so that the VTVs for that retention period are grouped on a common set of MVCs. After the migrations complete, you next run the `SWDIMS01` utility to create export MVCs to eject via ExLM for transport to the offsite vault. The VTVs on the export MVCs have expiry dates per the TMS (Zara). As these VTVs expire, you use an ExLM Custom Volume Report and the `SWDIMS02` utility to identify MVCs with no current VTVs in the offsite vault. You then use the Zara OFFSITE batch utility to create a list of these MVCs to return them to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs with a current data generation.
- **Catalog controlled** critical data sets are routed via a `TAPEREQ` to VTVs by a single Management Class that specifies `MIGRATE IMMEDIATE (DELETE)`. This Management Class specifies a two corresponding Storage Classes, one for onsite storage, one for offsite vaulting. After the migrations complete, you next run the `SWDIMS01` utility against the offsite Storage Class to create export MVCs to eject via ExLM for transport to the offsite vault. When ExLM reports MVCs with space utilization of less than a user-specified value, you use the `SWDIMS02` utility to identify these MVCs. You then "logically drain" these MVCs (by draining the onsite MVC), use the Zara OFFSITE batch utility to create a pull list of the MVCs eligible for reuse, and then return the offsite MVCs to the Production Site for reuse. The cycle continues with a fresh batch of export MVCs.

Caution: Please note the following gotcha. Some customers use HSC User Exit 06 to enter new tapes into the ACS and define them as scratch. This is exactly what we'd want for Nearline tapes, but scratch MVCs are a **not** a value add. So if you're entering MVCs and HSC User Exit 06 is set to define newly entered tapes as scratch, use the HSC `UEXIT` command to disable the exit before entering tapes to be used as MVCs.

"Offsite Vaulting - The Pickup Truck Access Method for Zara" consists of the following subsections:

- "Normal Operations" on page 121, where we show what the Production Site looks like when the data is flowing smoothly. Here, we also tell how to set up the system to withstand a failure.
- "Business Continuance" on page 137, which is how to stay open for business if the Production Site takes a major hit.
- "Business Resumption" on page 140, which is how to get back to normal operations once the Production Site is up and running again.

Normal Operations uses the StorageTek VSM Vault Utilities (the `SWDIMS01` and `SWDIMS02` utilities), so before diving into Normal Operations, let's turn the page and install these utilities.

Normal Operations

On page 119, we gave an overview of the system, disaster recovery needs, and operations to meet those needs. Figure 63 and Figure 64 on page 122 show the system during normal operations for VTV migration and recall. Figure 65 on page 122 shows the export and physical transport to the offsite vault of the MVCs that contain the critical data. There is, as we said, Some Assembly Required, so let's turn to "Migration, Export, Offsite Vault, and MVC Reuse" on page 123 for the details.

Normal Operations - VTV Migration

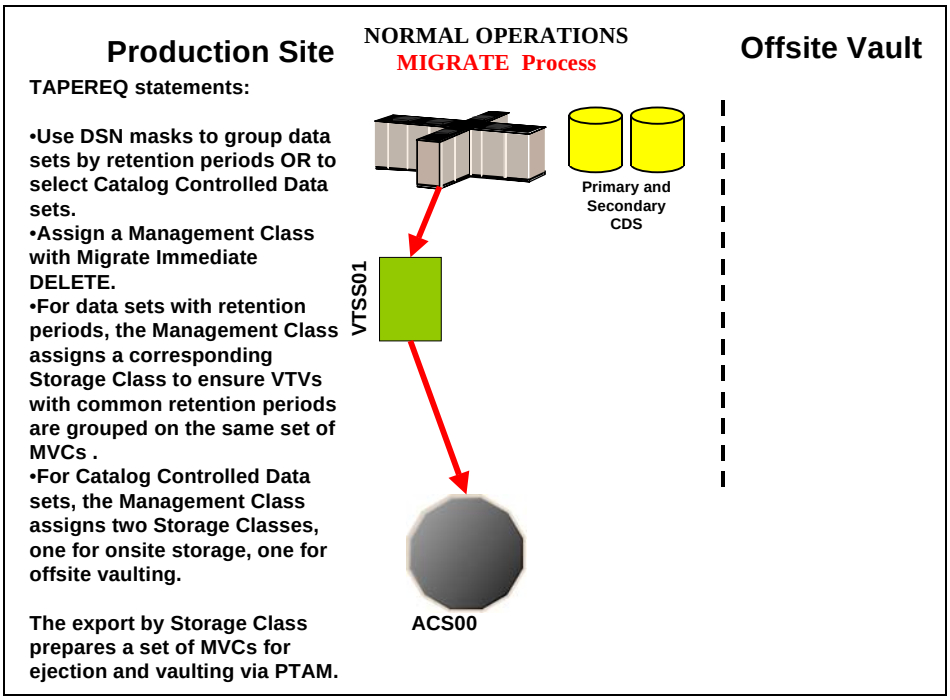


Figure 63. Normal Operations - VTV Migration

Normal Operations - VTV Recall

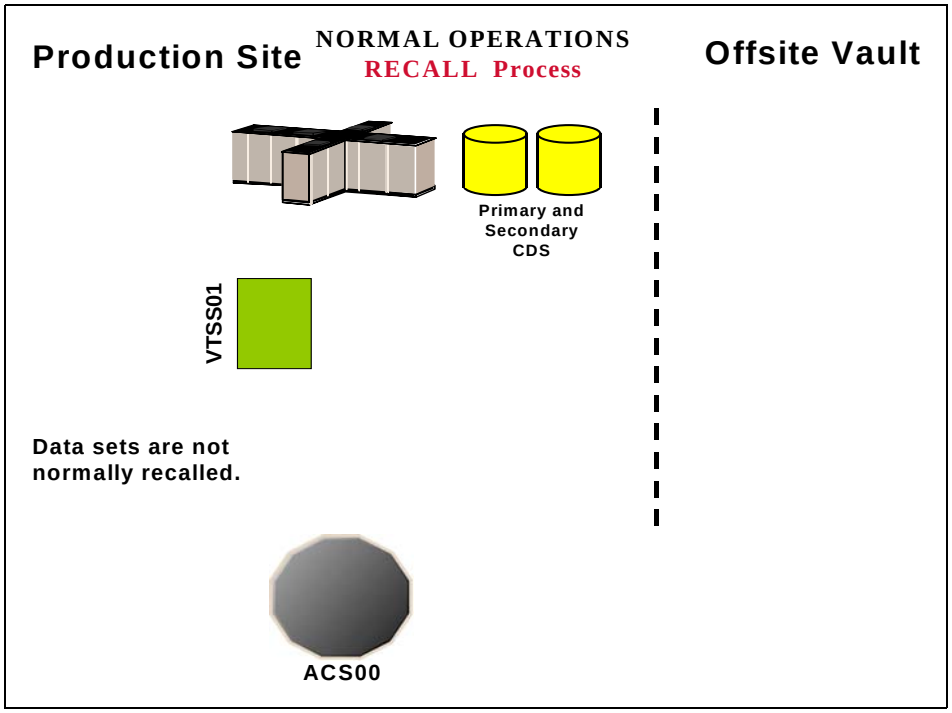


Figure 64. Normal Operations - VTV Recall

Normal Operations - MVC Export

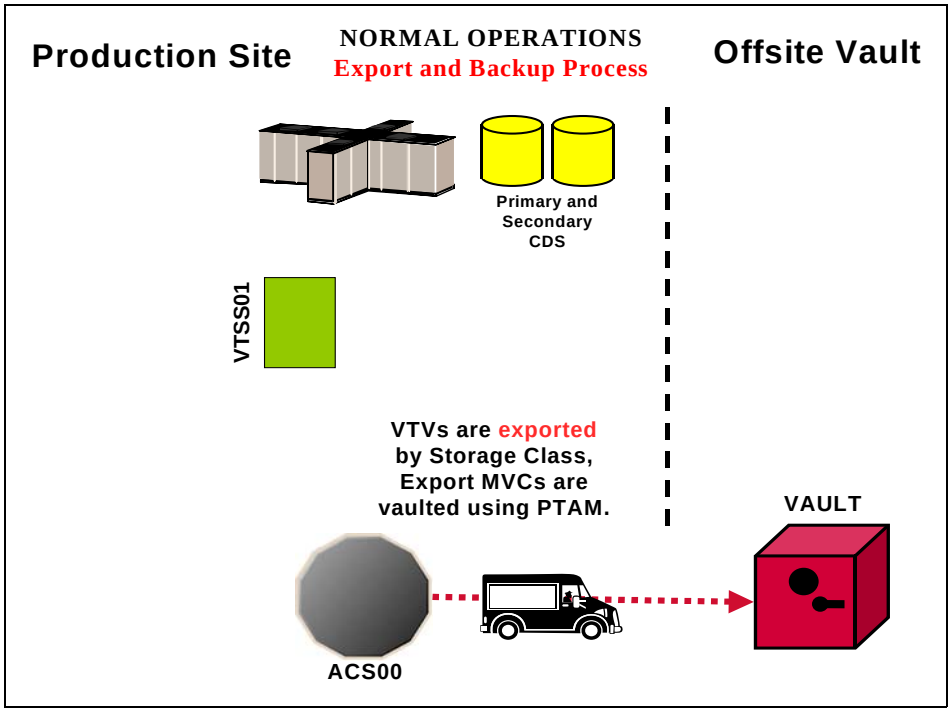


Figure 65. Normal Operations - MVC Export

Migration, Export, Offsite Vault, and MVC Reuse

In “Normal Operations” on page 121, we showed the system under normal operations. This section tells how to route critical data **in data sets with retention periods** and **catalog controlled data sets** to VTVs, migrate them to a common set of MVCs, and vault the MVCs offsite.

Table 20 on page 124 and Table 19 on page 124 are about two distinctly different kinds of data sets:

- First, in Table 19 on page 124, for **catalog controlled data sets**, we’re again showing the use of a data set name mask on TAPEREQ. As above, use other data set characteristics that you specify on TAPEREQ if that works better. Note that with catalog controlled data sets, the TAPEREQ specifies a single Management Class that duplexes the data via two Storage Classes. This produces an offsite MVC and an onsite MVC with the critical data, which, as we’ll see later, is critical to reusing and recycling MVCs.
- In Table 20 on page 124, for **data sets with retention periods**, we’re showing a situation where you have multiple data set name masks, each of which corresponds to a different retention period for the data sets selected by that mask. The fewer of these, the better (read what the VTCS documentation says about Too Many Storage Classes). That is, for *each* data set name mask you need a corresponding Management Class and Storage Class.

  With TAPEREQ, in fact, you have lots of choices...for example, you can key on jobname or stepname, if that works better. If you want to key on retention periods, data set name is probably the right move, however. We just want to make sure you know that you **do** have the flexibility to key on different data set groupings with different characteristics.

So at this point your question is “What do I do if I hav***e both*** flavors of data sets?” Well, we thought of that, too, because in the real world this is a very likely situation. Back in the MVC reuse section, which is one of the key areas to get right, we break that part of the process down for you. You can read about all of that in “Returning MVCs from the Offsite Vault for Reuse” on page 131. Another useful tool, however, is the overall process flow for “mixed” shops that we show in “A Sample Offsite Vaulting Work Flow” on page 173.

- In this example, we're selecting the MVCs for offsite vaulting from a Named MVC Pool. This is optional, but it's not a bad idea because:
  - All the non-critical jobs select MVCs either from another Named MVC Pool or from the overall MVC Pool (except for the Named Pool used for offsite vaulting).
  - At the Recovery Site, you only have to define the MVCs in the Named Pool.

  Once again, however, read the *caveats* in the VTCS documentation about the potential gotchas with Named MVC Pools.
- Finally, throughout the procedures, we refer you back to Table 20 or Table 19. In fact, what we *really* mean is your versions of these tables, which you'll have to create to match your shop's situation...as recorded in "Offsite Vaulting Configuration Record" on page 177.

Table 19. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Catalog Controlled Data Sets without Retention Periods

Management Class	Storage Classes	Use	Named MVC Pool	TAPEREQ Data Set Mask
mgmtcat	storon storoff	On site MVCs Offsite MVCs	poolcat	maskcat

Table 20. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Data Sets with Retention Periods

Management Class (mgmt-class-name)	Retention Period RETPD(LT,n)	Storage Class (stor-clas-name)	Named MVC Pool (poolname)	TAPEREQ Data Set Mask (mask)
mgmtret1	1 to 7 days	storret1	*poolret1*	maskret1
		. . .		
mgmtretn	Greater than one century	storretn	*poolretn*	maskretn

- Let's not forget the SWDTMS01 and SWDTMS02 utilities, which we'll use in the following sections. For some reference information, see "SWDTMS01 Utility" on page 144 and "SWDTMS02 Utility" on page 155.

  The whole of Normal Operations takes some doing, so let's break it down into the following sections:

  - "Setting Up the System" on page 125
  - "Creating the Export MVCs and Vaulting Them Offsite" on page 129
  - "Returning MVCs from the Offsite Vault for Reuse" on page 131

Setting Up the System

To set up the system:

1. **First, we'll enable the Advanced Management Feature.**

   The Advanced Management Feature is required to implement Storage Classes, run the EXPORT and IMPORT utilities, and use selected Management Class parameters (such as MIGPOL).

2. **Let's build the Management Classes that assign a corresponding Storage Class and do an immediate migrate with delete.**

   - **For data sets with retention periods**, *each* of the MGMTclas statements (you create a separate statement for each retention period) looks something like Figure 66:

```
MGMT NAME(mgmtretn) IMMED(DELETE) MIGPOL(storretn) DELSCR(YES)
```

Figure 66. Management Classes for VTVs with Critical Data

   - **For catalog controlled data sets**, the MGMTclas statement looks something like Figure 67:

```
MGMT NAME(mgmtcat) IMMED(DELETE) MIGPOL(storon,storoff) DELSCR(YES)
```

Figure 67. Management Classes for VTVs with Critical Data - Catalog Controlled Data Sets

   In Figure 66 and Figure 67, each Management Class specifies immediate migrate delete, which is designed to immediately put VTVs in this Management Class on the migration queue and delete the VTSS copy once it is migrated. This ensures quick migration and frees the VTSS buffer. We also assign the corresponding Storage Class to the MVCs that contain the migrated VTVs as shown in Table 20. on page 124 or Table 19 on page 124.

Warning: Note that for both types of data sets, the Management Class specifies DELSCR(YES). If you haven't already done so, **go back and read** the planning information in "DELSCR Considerations" on page 6.

3. **Next, we'll create the Storage Classes that own the MVCs that contain the migrated VTVs.**

   - **For data sets with retention periods**, each of the STORclas statements (you create a separate statement for each retention period) looks something like Figure 68:

```
STOR NAME(storretn) MVCP(poolret)
```

Figure 68. Storage Classes for Data Sets with Retention Periods

   For the Storage Class names to plug in, see Table 20. on page 124. In Figure 68, the STORclas statement associates each Storage Class with the specified Named MVC Pool.

   - **For catalog controlled data sets**, the STORclas statements looks something like Figure 69:

```
STOR NAME(storon) MVCP(poolcat)
STOR NAME(storoff) MVCP(poolcat)
```

Figure 69. Storage Classes for Catalog Controlled Data Sets

   For the Storage Class names to plug in, see Table 19. on page 124. In Figure 69, the STORclas statements associate each Storage Class with the specified Named MVC Pool.

4. **Next, we'll use the MGMTDEF command to load the MGMTclas and STORclas control statements we created in Step 2 on page 125 and Step 3 on page page 126.**

5. **Now let's create TAPEREQ statements to route the critical data to VSM and assign the corresponding Management Class to the data.**

   - **For data sets with retention periods**, each of the TAPEREQ statements (you create a separate statement for each retention period) looks something like Figure 70:

```
TAPEREQ DSN(maskretn) MEDIA(VIRTUAL) MGMT(mgmtretn) RETPD(LT,n)
```

Figure 70. TAPEREQ Statements for Data Sets with Retention Periods

Note: The RETPD parameter is not supported under JES3, so use another TAPEREQ data set selection method (for example, JOBNAME).

   See Table 20. on page 124 for the correct values to fill in for Figure 70.

   - **For catalog controlled data sets**, the TAPEREQ statement looks something like Figure 71:

```
TAPEREQ DSN(maskcat) MEDIA(VIRTUAL) MGMT(mgmtcat)
```

Figure 71. TAPEREQ Statement for Catalog Controlled Data Sets

   See Table 19. on page 124 for the correct values to fill in for Figure 71.

6. **Use the TREQDEF command to load the TAPEREQ control statements we created in Step 5.**

7. **If they aren't already in place, set up the Zara vault codes for the exported MVCs.**

   Zara uses the data set name (and optionally the name of the job that creates the data set) to assign vaulting codes. For more information on setting up vaulting codes to achieve the desired vaulting assignments and rotation, see OFF-SITE Storage Processing in *Zara User's Guide*.

   The name of the data set that the SWDIMS01 utility assigns to the MVCs in the Zara Database is:

   DSN=prefix.storclas.MVCTAPE.RESERVED

   Where:

   - prefix is the HLQ specified in the MvcDsnPreFiX parameter of the STORclas statement. For more information, see "EXPiryPeRioD Statement Example" on page 168.
   - storclas is a Storage Class created in Step 3 on page 126.

   For example, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and LE8DAYS for the Storage Class for retention periods of less than 8 days, the SWDIMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.LE8DAYS.MVCTAPE.RESERVED

   Similarly, if you specified a prefix of SYS1.VSMDR on the MvcDsnPreFiX parameter and CATOFF for the Storage Class for "offsite" MVCs with catalog controlled data sets, the SWDIMS01 utility assigns to the MVCs for that Storage Class the following DSN:

   SYS1.VSMDR.CATOFF.MVCTAPE.RESERVED

8. **There is one last item...**

   ...which is defining the MVCs to the TMS. As it says in the VTCS documentation:

   > "Access to the MVCs via an RTD bypasses the MVS intercepts put in place by the tape management system so that it does *not* record within its database any access to the MVCs by VSM and does *not* automatically provide protection against inadvertent overwrites of non-expired data on MVCs. Therefore, if you choose to define MVCs to the tape management system, StorageTek **strongly recommends** that you define them as non-scratch, non-expiring volumes."

   Well, that isn't going to work so well for MVCs that you want to reuse, so we have to find another way. So if you want to define the MVCs to Zara and adequately protect them, try this:

   a. Do a format extend to add the MVC volsers to the Zara Database.

   b. Second, define the MVCs as a scratch subpool to Zara.

   c. Finally, write a rule that says that *only* HSC/VTCS can write to the MVCs as scratches.

   Section 1, Setting Up the System, is done, so let's proceed to "Creating the Export MVCs and Vaulting Them Offsite" on page 129.

To create the export MVCs and vault them offsite:

1. **Ensure that all VTVs with critical data are migrated.**

   The TAPEREQ statements in Step 5 on page 126 are the triggers to start the VTV migrations to the MVCs we'll export. But how do we know when all the VTVs with critical data are migrated? Basically, we write a "watchdog" program that monitors VTV migration and reports when the *last* critical VTV is migrated.

2. **Run the SWDIMS01 utility.**

   The SWDIMS01 utility, working together with the JCL you create to invoke it, does the following:

   - **Runs the EXPORT utility** against **all** Storage Classes we created in Step 3 on page 126 **and creates a comprehensive Manifest File** that includes all these Storage Classes. We use the STORclas statement of the parameter file to specify one or more Storage Classes. For more information, see "EXPiryPeRioD Statement Example" on page 168.

     Note: If an MVC in a specified Storage Class is in use when the SWDIMS01 utility runs, it will be skipped and will not be exported until the next time the utility runs.

     The comprehensive Manifest File is used, if needed, in Business Continuance mode (see Step 4 on page 137)...so it's probably a good idea to have one copy of the file on disk and another on tape stored offsite.

   - **Creates an MVC update file**, which is used as input to ZARAUTL in Step 3 on page 130, to set data set names and expiration dates for the MVCs to be sent offsite. These commands set the data set expiration date to PERMANENT.

   - **Creates a VTV update file**, which can optionally be used as input to ZARAUTL in Step 3 on page 130, to change the TMS VTV records.

     These control cards reflect the state of the VTVs being sent offsite, and it's probably not a good idea to alter them. If, for some reason, you have a need to change the state of VTVs being sent offsite, contact StorageTek Software Support for information on the use of these cards.

Hint: Good Practices suggest that we **only** want to vault volumes that are resident in an ACS. To do this, run the SWDIMS01 utility with the STORCLAS ALL(OFF) parameter of the parameter file. If, however, you want to see a selection list of all volumes that **should** be vaulted, whether they are ACS resident, use the STORCLAS ALL(ON) parameter. For more information, see "STORCLAS" on page 169. **Note that**, even if you specify the ALL(OFF) parameter, the Manifest File is **always** "comprehensive."

For example JCL for the SWDIMS01 utility, see Figure 78 on page 146 and Figure 79 on page 147.

3. **Next, we run ZARAUTL to update the Zara Database with MVC information.**

   ZARAUTL uses the input from Step 2 on page 129 to update the MVC volume records in the Zara Database with a dummy data set name, expiration date, creation date, and so forth.

Hint: In Step 2, if the SWDIMS01 utility **does not** select any MVCs for update, SWDIMS01 ends with a return code of 4. Because the Zara Database update in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS01 runs successfully, selects MVCs, and completes with a return code of 0 as shown in Figure 80 on page 148.

4. **To assign the vault codes and rotation to the MVCs, run the Zara OFFSITE batch utility.**

   ...where we set up the vault codes in Step 7 on page 127 and updated the Zara Database with MVC information in Step 3 on page 130.

   For more information on the Zara OFFSITE Utility, see OFF-SITE Storage Processing in *Zara's User Guide*.

5. **Keying on the Vault Code from Step 7 on page 127, we use ExLM to eject the MVCs from the ACS.**

   Figure 81 on page 149 shows example JCL to use ExLM to eject the MVCs.

6. **Finally, we use PTAM (Pickup Truck Access Method) to do the offsite vaulting of the MVCs we ejected in Step 5.**

   Because we have a finite number of MVCs, when VTVs go non-current on exported MVCs, we need a way to cycle these MVCs back into the system for reuse. For that procedure, see "Returning MVCs from the Offsite Vault for Reuse" on page 131.

Returning MVCs from the Offsite Vault for Reuse

In “Creating the Export MVCs and Vaulting Them Offsite” on page 129, we migrated VTVs with critical data to MVCs and exported those MVCs to the Offsite Vault. We don’t have an infinite number of MVCs for vaulting, so we need a way to recycle vaulted MVCs back into the system when some or all VTVs on these MVCs go non-current. And there are three flavors, so pick the one that works for your shop and follow the referenced procedure:

1. **Flavor 1 - Sites with *only* data sets with retention periods**, when ExLM reports that all VTVs on an MVC have reached their expiry dates in Zara, the current VTV count goes to zero and we can return the MVC to the Production Site for reuse. For procedures, see “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 135.

2. **Flavor 2 - For sites with *only* catalog controlled data sets**, we use an ExLM report to look for vaulted MVCs that have a usage percentage less than a percentage we specify. In other words, ExLM looks for MVCs that have enough free space to make it worthwhile to reuse these MVCs. What’s a reasonable percentage to specify? The answer, of course, is “it depends on the needs of your shop.” Specify a high percentage and you’ll free up lots of MVCs...at the cost of high VSM/system activity. Specify a low percentage and you’ll have lots of fragmented MVCs in the offsite vault. Try starting with 25% and adjust as needed.

   The way we reuse these MVCs is...well, it’s complex, so let’s slow down and see how it works:

   1. MVCs with less than `n` percent usage are marked `LOST`. Now why would I do that? Because I have an equivalent on site copy and I don’t want to put an operator through mounting the offsite MVCs to drain it, see number 2.

   2. I try to run a drain against the offsite MVC, it’s lost, so instead VTCS “logically drains” the offsite MVC by recalling all its VTVs from the MVC copy that is onsite. Because the recalled VTVs have the DR Management Class, they are migrated to a new MVC that can be taken offsite. The new MVC is now the input to Step 2 on page 129, which detects MVCs with Storage Classes that require offsite vaulting.

      So VTCS “logically drained” the offsite MVC, which is goodness because the offsite MVC is now 0% full. It is now eligible for selection by Step 1 on page 135 and is returned per Step 7 on page 136.

   And for the step-by-step implementation of this process, see “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 133.

Note that you are not done until, as it says in Step 4 on page 134, you next complete the procedure described in “Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods” on page 135...because the logical draining has created MVCs with a zero current VTV count. Is that cool, or what?

Hint: How often do you want to do this defragmentation? The answer is “it depends on the needs of your shop and your level of vaulting/reuse activity”...but doing this once a week is probably a good starting point.

3. **Flavor 3 - For "mixed" sites with *both* catalog controlled data sets *and* data sets with retention periods,** you do the same thing (and for the same reasons) that you do for sites with only catalog controlled data sets. That is, you first do the procedure in "Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets" on page 133, followed by the procedure in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 135.

Note: Each of the three procedures described above **requires** the standard `OFFSITE` update process described in *Zara User's Guide*.

Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets.

To Recycle Offsite MVCs That Contain VTVs With Catalog Controlled Data Sets:

1. **First, let's run an ExLM Custom Volume Report...**

   ...to find MVCs with less than a specified percent used.

   Figure 82 on page 157 shows a JCL example that generates an ExLM report that selects vaulted MVCs with less than a specified percent used. In Figure 82 on page 157:

   - `DSN=prefix.storoff.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the Zara Database, as described in Step 7 on page 127.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "EXPiryPeRioD Statement Example" on page 168.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specify in Step 1 on page 135. The difference is in the *storclas* value for the middle qualifier on the DSN. In this step, it is the single DSN for the data set assigned to all MVCs that contain VTVs with catalog controlled data sets. In this step, you want to drain ***only*** the MVCs with catalog controlled data sets, not the MVCs with data sets with retention periods. If you go all the way back to Step 3 on page 126, you'll see that it is, in fact, `storoff`, which is the Storage Class for the offsite (vaulted) MVCs.

   - `loccode` is the location code for the offsite vault.
   - `percent` is the specified percentage use. See page 131 for a discussion of what constitutes a reasonable value.
   - `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 84 on page 159 shows example JCL where the `SWDIMS02` utility:

   - Creates the appropriate `MVCMAINT` commands to mark as `LOST` the MVCs selected in Step 1.
   - Creates the appropriate `MVCDRAIN` commands to make all the swell stuff happen that we described starting in number 1. on page 131.

3. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 133 to enable draining the fragmented offsite MVCs.**

Caution: Here we need to take a look at Figure 84 on page 159. This is the JCL to run the SWDIMS02 utility in Step 2 on page 133. Note that the JCL specifies a DD that contains "simplexed" output cards (SWDSIMP) and a DD that contains "duplexed" output cards (SWDDUP). In this step, you **must** run MVCMAINT against the duplexed cards (SWDDUP)!

Also note that, in Step 2, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in Step 3 will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

Figure 85 on page 160 shows example JCL to run SWSADMIN against the duplexed cards only if MVCs are selected.

Finally, **Also note that**...we're running a drain to recall the current VTVs from the MVC...remember that these are MVCs that are x percent used. How does this actually happen? For the long version, see page 131. Here's another data point:

> After Step 3 completes, the still current VTV, which had one copy on the offsite MVC which was drained, will have one copy on the onsite MVC, and a new copy on another onsite MVC which has the same Storage Class as the original offsite MVC. In the meantime, the offsite MVC will have had its storage class removed. This new onsite DR MVC will be picked up in the next SWDIMS01 run, and will be sent offsite.

And, additionally:

> The drain process also clears the LOST status of the offsite MVC.

4. **Don't stop now...**

...because, for all the good reasons we gave you back on page 131, this process **feeds into** and **requires** that you next complete the procedure described in "Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods" on page 135.

Recycling Offsite MVCs - VTVs with Data Sets with Retention Periods.

To Recycle Offsite MVCs That Contain VTVs With Data Sets with Retention Periods:

1. **First, let's run an ExLM Custom Volume Report to find any MVCs vaulted for data sets with retention periods with no current VTVs....**

   Figure 83 on page 158 shows a JCL example that generates an ExLM report that selects vaulted MVCs with no current VTVs. In Figure 83 on page 158:

   - `DSN=prefix.storclas.MVCTAPE.RESERVED` is the name of the data set that the `SWDIMS01` utility assigns to the MVC in the Zara Database, as described in Step 7 on page 127.

     `prefix` is the DSN prefix that you specify on the `MvcDsnPreFiX` parameter of the `STORCLAS` statement. For more information, see "EXPiryPeRioD Statement Example" on page 168.

     Caution: There is a subtle but **very critical difference** between the DSN you specify here and the DSN you specified in Step 1 on page 133. The difference is in the *storclas* value for the Second Level Qualifier on the DSN. In this step, we want to detect all offsite MVCs with 0 VTVs. Therefore, the DSN mask in the ExLM job should be "`prefix.**`".

   - `loccode` is the location code for the offsite vault.

   `SWDEXLST` is the required DDname of the data set that contains the output of the ExLM report; see "SWDEXLST" on page 155 for more information.

2. **Run the `SWDIMS02` utility to process the flat file report generated in Step 1.**

   Figure 86 on page 161 shows example JCL where the `SWDIMS02` utility:

   - Creates the TMS `ZARAUTL` commands to modify TMS volume records for MVCs. These commands set the Expiration Date specified on the `EXPIRYPERIOD` statement to allow a return of the MVC from the vault. For more information, see "EXPIRYPERIOD" on page 167.
   - Creates the appropriate `MVCMAINT` commands to reset the Readonly bit to make these MVCs available for reuse after they're reentered into the ACS.
   - Creates the appropriate `MVCDRAIN` commands to...trust us, it's a good idea, and we'll explain it all in Step 4 on page 136.

3. **Run the `ZARAUTL` procedure to process the control cards generated in Step 2...**

   ...to update the Zara Database accordingly. We do this, together with some other fun `SWSADMIN` stuff, as shown in the example in Figure 87 on page 162.

4. **Run the SWSADMIN program to process the MVCMAINT and MVCDRAIN commands created in Step 2 on page 135 to reset the readonly status of the MVCs that will be returning to the production site...**

   ...and to also do a logical drain on those MVCs. “Now why,” you ask yourself, “Is he telling me to drain an *empty* MVC?” And the answer is that invoking MVCDRAIN does more than just draining VTVs, it also affects an MVC state change, which in this case is a Good Thing. Specifically, the MVDRAIN operation removes the Storage Class from the MVC, which prevents the MVC from being selected when SWDIMS01 is exporting MVCs...and also makes the MVC available to the universe, not just that Storage Class. Don’t worry, these empty MVCs will not have to be mounted to be logically drained.

   Caution: Here we need to take a look at Figure 86 on page 161. This is the JCL to run the SWDIMS02 utility in Step 2 on page 135. Note that the JCL specifies a DD that contains “simplexed” output cards (SWDSIMP) and a DD that contains “duplexed” output cards (SWDDUP). In this step, you **must** run MVCMAINT against the simplexed cards (SWDSIMP)!

   Also note that, in Step 2 on page 135, if the SWDIMS02 utility does not select any MVCs for update, SWDIMS02 ends with a return code of 4. Because the SWSADMIN commands in this step will fail if no MVCs are selected, StorageTek recommends that you code this jobstep to run only if SWDIMS02 runs successfully, selects MVCs, and completes with a return code of 0.

   Figure 87 on page 162 shows example JCL to run SWSADMIN against the simplexed cards only if MVCs are selected.

5. **Run the Zara OFFSITE utility batch job(s) to create the list of MVCs to be returned from offsite storage for reuse.**

6. **Using OFFSITE picklists, do the PTAM thing to take the MVCs ejected in Step 5 on page 130 to the offsite vault.**

7. **While you’re at the offsite vault...**

   ...pick up any MVCs which are to be moved back to the onsite library. This is a little hard to visualize, so think of it this way: We actually identified these MVCs in Step 2 on page 135 during **a previous run** of this procedure that occurred **greater than** nn **days ago**, where nn is the expiry period specified on the EXPIRYPERIOD statement. For more information, see “EXPIRYPERIOD” on page 167.

   Great, we’re done...that is, until the next time we get to cycle back into the defrag procedure described in “Recycling Offsite MVCs - VTVs with Catalog Controlled Data Sets” on page 133.

Business Continuance

Okay, the Unthinkable has occurred, and the Production Site has taken a major hit. Now what? Well, the operations staff next wants to switch to business continuance mode. Note that because of the setup we did in “Normal Operations” on page 121, we can quickly and effectively resume operations because:

- All critical data is migrated and vaulted.
- A Recovery Site (which could be a vendor such as Comdisco) is standing by with VSM installed.

Because we did our homework on the setup, the switch to Business Continuance Mode at the Recovery Site is quick and straightforward via the MVC import shown in Figure 72 on page 138. After we switch to Business Continuance Mode, operations look like Figure 73 on page 138 and Figure 74 on page 139. We’ll use the following procedure to switch to Business Continuance Mode.

To switch to Business Continuance Mode at the Recovery Site, do the following:

1. **Create a new CDS at the Recovery Site.**

   This CDS reflects LIBGEN and VTCS configuration at the Recovery Site; we’ll populate this with VTV and MVC information in Step 4.

2. **Enter only critical MVCs into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

3. **Start an HSC audit on the panels we filled in Step 2.**

4. **Using the *most current* comprehensive Manifest File, run an import into the *new* CDS we created in Step 1.**

   We created the comprehensive Manifest File in Step 2 on page 129.

   Hint: When the import completes, we can start work using the critical data sets.

5. **Enter all remaining MVCs (and any standard Nearline volumes) into the Recovery Site ACS.**

   Fill complete rows, one panel at a time.

6. **Start an HSC audit on the panels we filled in Step 5.**

7. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Recovery Site VTSSs.**

8. **Now, and only now, start sending non-critical data to the Recovery Site VTSSs.**

   This completes this procedure, and we’re now up and running again. As soon as the Production Site is back in operation, run, do not walk, to “Business Resumption” on page 140.

Business Continuance - MVC Import

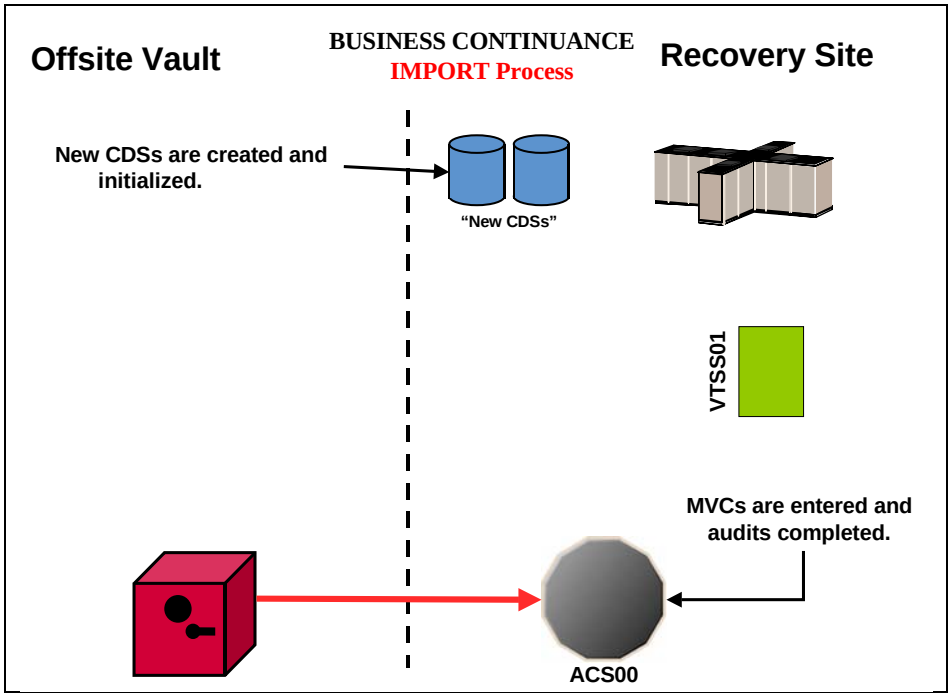


Figure 72. Business Continuance - MVC Import

Business Continuance - VTV Recall

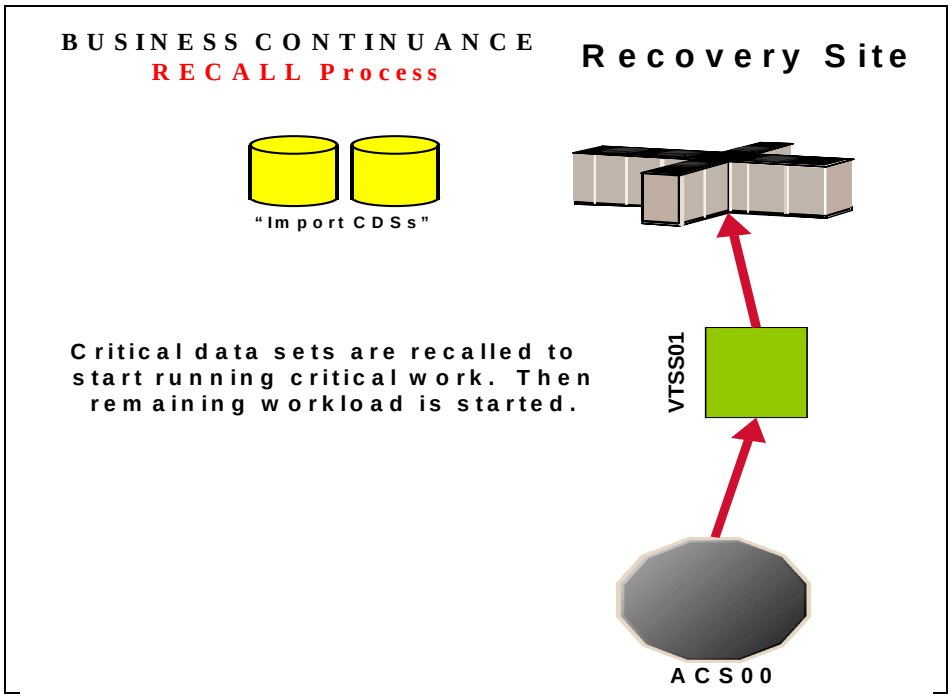


Figure 73. Business Continuance - VTV Recall

Business Continuance - VTV Migration

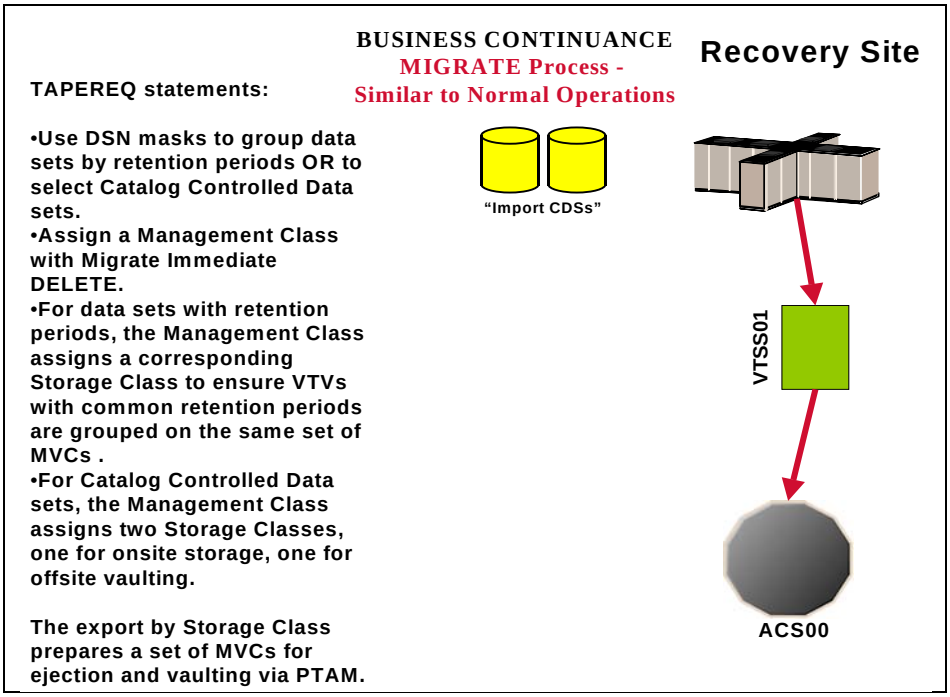


Figure 74. Business Continuance - VTV Migration

Business Resumption

The Production Site is back up as a data center, so it's time to get back to business as usual as shown in Figure 75 on page 141, Figure 76 on page 141, and Figure 77 on page 142.

The switch from Business Continuance to Business Resumption should look familiar, because it's very similar to what we did in "Business Continuance" on page 137...only this time we're using information and resources from the Recovery Site to recreate the Production Site VSM operation.

To resume normal operations:

1. **Clean the Production Site VTSSs.**

   This procedure, which is done by StorageTek hardware service, wipes out any extraneous VTVs that may be lingering around.

2. **Run an Export against all MVC ranges used at Recovery Site, using the Recovery Site CDS.**

   Note: At this point, we're done with the Recovery Site...although we might want to keep it up and running until we reach Step 10.

3. **Create a new CDS at the Production Site.**
4. **Enter only critical MVCs into the Production Site ACS.**

   Fill complete rows, one panel at a time.

5. **Start an HSC audit on the panels we filled in Step 4.**
6. **Using the Manifest File from Step 2, run an import into the *new* CDS we created in Step 3.**

   Hint: When the import completes, we can start work using the critical data sets.

7. **Enter all remaining MVCs (and any standard Nearline volumes) into the Production Site ACS.**

   Fill complete rows, one panel at a time.

8. **Start an HSC audit on the panels we filled in Step 7.**
9. **When the audits and the import complete, consider prestaging data sets with requirements for quick access to the Production Site VTSSs.**

   Note: After the VTVs that contain critical data sets are VTSS resident, you must enable **access** to the data sets on these VTVs, either by recataloging them, or using JCL statements such as `VOL=SER=vtvnnn`.

10. **Now, and only now, start sending new data to the Production Site VTSSs.**

Business Resumption - VTV Migration

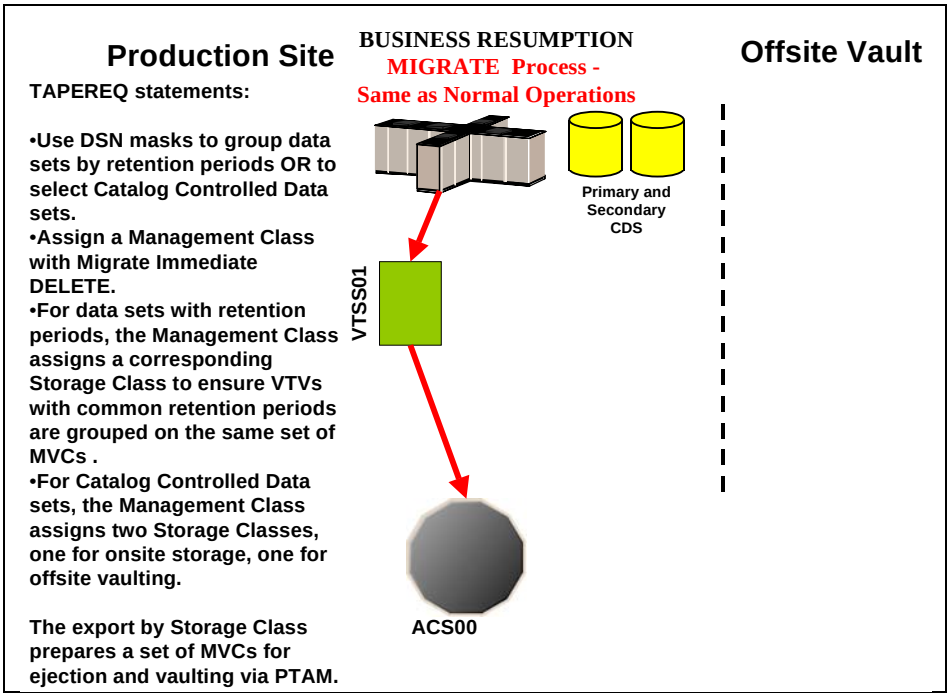


Figure 75. Business Resumption - VTV Migration

Business Resumption - VTV Recall

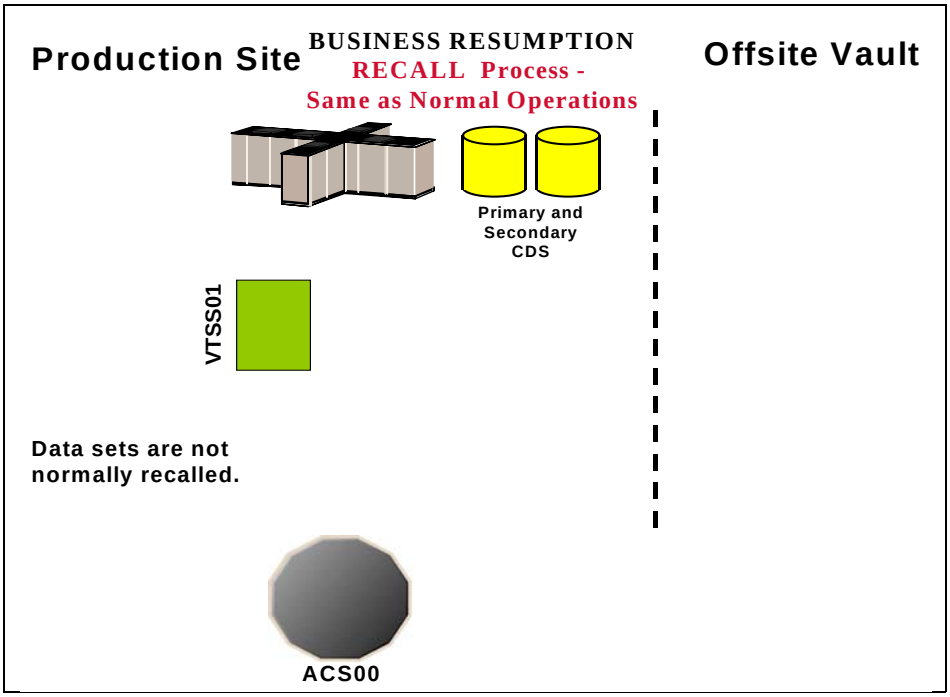


Figure 76. Business Resumption - VTV Recall

Business Resumption - MVC Export

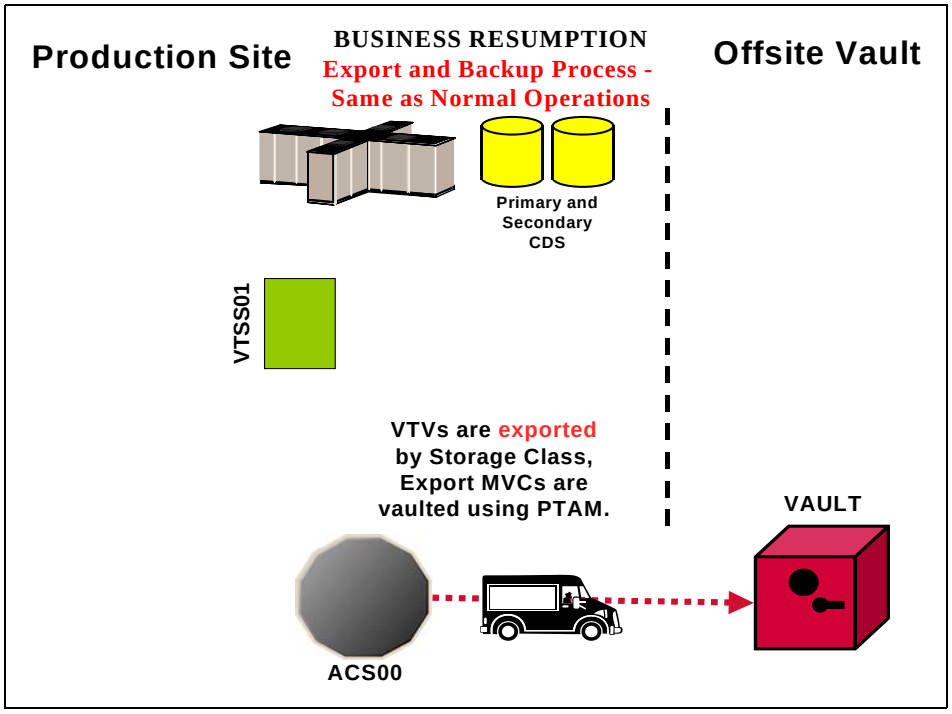


Figure 77. Business Resumption - MVC Export

The VSM Vault Utilities

This section contains reference information for the following utilities:

Note: The Common Parser Tool runtime module (SKQPARSE) must be added to the STEPLIB that contains the utilities.

SWDTMS01 Utility

The SWDTMS01 utility generates a VTCS EXPORT Manifest File for the specified Storage Classes and generates TMS update commands to allow vaulting of the export MVCs.

Parameters

None.

Usage

You must specify a Storage Class on the STORCLAS statement. For more information, see "STORCLAS" on page 169.

SWDTMS01 JCL Requirements

STEPLIB

In addition to the VSM Vault Utilities load library, it must include the libraries containing the VTCS utility load modules unless these are in the linklist concatenation. All libraries in STEPLIB must be APF authorized.

SLSCNTL

specifies the HSC primary CDS data set.

SWDMIN

Specifies a sequential work data set with attributes RECFM=VB, LRECL=75, and BLKSIZE=31030. This dataset must be large enough to contain all exported MVCs, where each block stores approximately 410 MVC records. You can allocate it as a temporary data set because its contents do not need to be retained.

SWDVIN

Specifies a sequential work data set with attributes RECFM=VB, LRECL=52, and BLKSIZE=31030. This data set must be large enough to contain all reported VTV volumes, where each block stores approximately 550 VTV records. You can allocate it as a temporary data set because its contents do not need to be retained.

SLSPRINT

Specifies a sequential work data set with attributes RECFM=FBA, LRECL=121, and BLKSIZE=23474. This data set must be large enough to contain all exported MVC volumes, where each block stores approximately 160 MVC records. You can allocate it as a temporary data set because its contents do not need to be retained.

SLSIN

Specifies a single track sequential work data set with attributes RECFM=FB, LRECL=80, and BLKSIZE=9040. You can allocate it as a temporary data set because its contents do not need to be retained.

SWDPRM

Specifies the parameter file used with the SWDTMS01 utility. The parameter file can be a PARMLIB member. For more information, see "VSM Vault Utilities Parameter Files" on page 166. This file is required because you must specify a STORCLAS statement in this file.

SWDMAN

Specifies the generated comprehensive Manifest File which must be retained for recovery processing. This data set must be large enough to allow for all export MVCs in the specified Storage Classes.

In the example in Figure 79 on page 147, this data set is allocated as a permanent file with a datestamp in the LLQ. If you use this technique, change the LLQ each time you run the utility.

SWDMOUT

Contains the generated TMS commands for the MVCs for offsite vaulting.

SWDVOUT

Contains the generated TMS commands for the VTVs. Not used at this time.

SWDRPT

Contains the VTCS SWSADMIN utility messages from the EXPORT, MVCRPT and VTVRPT utilities.

SYSPRINT

Contains the SWDTMS01 utility messages.

SYSTERM

Contains diagnostic messages. Normally empty.

SWDTMS01 JCL Example

Figure 78 on page 146, Figure 79 on page 147, and Figure 80 on page 148 show example JCL to run the SWDTMS01 and SWSADMIN utilities.

```
//jobname  JOB   MSGCLASS=o,REGION=2M,
//         CLASS=c,NOTIFY=userid
//*
//*********************************************************************
//* DELETE OUTPUT DATA SETS
//*********************************************************************
//STEP1    EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//DD01     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.MANIFEST.D2002023
//DD02     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.VTVUPDTE
//DD03     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.MVCUPDTE
//*********************************************************************
//* EXECUTE UTILITY 1 TO BUILD TMSUPDTE CONTROL CARDS
//*********************************************************************
//STEP2    EXEC PGM=SWDTMS01
//STEPLIB  DD  DSN=PROD2.VSMDR.LOAD,DISP=SHR
//         DD  DSN=PROD2.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=PROD2.HSC.V600.SLSLINK,DISP=SHR
//SLSCNTL  DD  DSN=PROD2.HSC.V600.DBASE1,DISP=SHR
//SLSCNTL2 DD  DSN=PROD2.HSC.V600.DBASE2,DISP=SHR
//SLSSTBY  DD  DSN=PROD2.HSC.V600.DBASE3,DISP=SHR
```

Figure 78. Example JCL for the SWDTMS01 Utility (Part 1)

```
//SWDPRM   DD  DSN=PREFIX.PARMLIB(PRMFILE1),DISP=SHR
//SWDMIN   DD  DSN=&MVCRPT,
//             DISP=(,DELETE,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(1,1)),
//             DCB=(RECFM=VB,LRECL=75,BLKSIZE=31030)
//SWDVIN   DD  DSN=&VTVRPT.
//             DISP=(,DELETE,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(1,1)),
//             DCB=(RECFM=VB,LRECL=52,BLKSIZE=31030)
//SLSPRINT DD  DSN=&MVCRPT,
//             DISP=(,DELETE,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(1,1)),
//             DCB=(RECFM=FBA,LRECL=121,BLKSIZE=23474)
//SLSIN    DD  DSN=&SLSIN,
//             DISP=(,DELETE,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(1,1)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDMAN   DD  DSN=VSM02.VSMDR.MANIFEST.D2002023,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(1,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
//SWDVOUT  DD  DSN=VSM02.VSMDR.VTVUPDTE,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(1,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDMOUT  DD  DSN=VSM02.VSMDR.MVCUPDTE,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(1,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDRPT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
/*
//
```

Figure 79. Example JCL for the `SWDIMS01` ***Utility (Part 2)***

```
//*******************************************************************
//*
//* THIS STEP CONDITIONALLY INPUTS UPDATE CARDS PRODUCED BY SWDTMS01
//*     AND UPDATES THE CATALOG FOR MVC RECORDS
//*
//*******************************************************************
//STEP3    EXEC PGM=TMSUPDTE,                        /* For CA-1
/*             PGM=CATINQR,                         /* For CA-TLMS
/*             CTMMUP,                              /* For CONTROL-M/Tape
/*             PGM=IKJEFT01,                        /* For DFSMSrmm
/*             ZARAUTL,                             /* For Zara
//         COND=(4,LE,STEP2)
//TMSRPT   DD   SYSOUT=*                             /* For CA-1
//SYSIN    DD   DSN=VSM02.VSMDR.MVCUPDTE,DISP=SHR    /* Not for DFSMSrmm
//SYSTSIN  DD   DSN=VSM02.VSMDR.MVCUPDTE,DISP=SHR    /* For DFSMSrmm
//SYSTSPRT DD   SYSOUT=*                             /* For DFSMSrmm
//*******************************************************************
```

Figure 80. Example JCL for the SWDTMS01 Utility (Part 3)

Note: If the PreserveMaximumVTCSReturnCode parameter is specified, then this step will need to be repeated with COND=(400,NE,STEP2) coded to ensure the TMS update statements are processed if the return code from the SWDTMS01 program is either 0 or 400. Alternatively, use the IF/THEN/ENDIF JCL statement construct instead of the COND parameter on the EXEC statement. This would allow testing for either 0 or 400 in the same step instead of having to have two steps.

ExLM JCL Example for SWDTMS01 - Ejecting MVCs for Vaulting

Figure 81 shows example JCL to run ExLM to eject the MVCs that were marked for vaulting by the SWDTMS01 utility.

```
//MM7047 JOB 434500,'MM7047',CLASS=A,NOTIFY=MM7047,
//       MSGCLASS=G,MSGLEVEL=1,REGION=64M
//*******************************************************************
//* KEY ON VAULT CODE ASSIGNED TO MVCS AND EJECT WITH APPROPRIATE MESSAGE
//*******************************************************************
//STEP1    EXEC PGM=LCMMAIN,PARM='PAGESIZE(45)',REGION=4M
//STEPLIB  DD DSN=PROD2.LCM.R600.LCMLINK,DISP=SHR
//         DD  DSN=PROD2.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=PROD2.HSC.V600.SLSLINK,DISP=SHR
//LCMTMSDB DD DSN=PROD2.yourtms.TMC,DISP=SHR,DCB=BUFNO=255
//LCMMSGS  DD SYSOUT=*
//SORTMSGS DD SYSOUT=*
//LCMLOG   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//LCMPARM  DD *
 OPTIONS
     NOSYNC
     EJCAP(000)
     EJMODE(SINGLE)
     NONSWAPPABLE
     TITLE('VSM DR MVC EJECTION RUN     ');
 TMS
    yourtms
    NAME('YOUR TMS');
 METHOD
    NAME(VAULT)
    EJECT(YES);
 LOCATION
    CODE(VDR1)
    METHOD(VAULT)
    EJCAP(000)
    EJMODE(SINGLE)
    SLOTS
    EJMSG('VSM DR MVCS SEND OFFSITE');
 MANAGE
    ACSID(00);
/*
//
```

Figure 81. Example JCL for ExLM to Eject MVCs for Vaulting

In Figure 81, we're assuming that offsite slots are assigned in CA-1; if not, specify NOSLOTS.

Return Codes

Table 21 describes return codes from the SWDTMS01 utility.

Table 21. SWDTMS01 Return Codes

Return Code	Description
0	Utility ran successfully and selected MVCs to process.
4	Utility ran successfully but did not select MVCs to process.
8	Utility terminated due to an error condition.

Note: The above return codes may be modified by the specification of "PreserveMaximumVTCSReturnCode" on page 170.

Messages

Messages are numbered in the format SWDnnnnT where T is I (information), W (warning) or E (error). Warnings produce a return code 4, while errors produce a return code 8.

SWD1000E Unable to open SWDMIN DD: ddname

Explanation: The input DD for the MVC report could not be opened. Additional information on the error may be provided.

SWD1001E Unable to open DD SWDMOUT: ddname

Explanation: The output DD for the MVC TMS update commands could not be opened. Additional information on the error may be provided.

SWD1002E Unable to open SWDVIN DD: ddname

Explanation: The input DD for the VTV report could not be opened. Additional information on the error may be provided.

SWD1003E Unable to open SWDVOUT: ddname

Explanation: The output DD for the VTV TMS update commands could not be opened. Additional information on the error may be provided.

SWD1004W Invalid storage class found for MVC mmmmm on line nn, skipping

Explanation: The Storage Class name found in the MVC work file for the MVC mmmmm is not a valid MVC name. The MVC is not included in further processing.

SWD1005W No VTSS found for MVC on line line, skipping

Explanation: No VTSS name found in MVC report for an MVC. The MVC is skipped.

SWD1006E Error reading vtv report at line line: status

Explanation: There was an error reading the VTV report. Additional information on the error may be given.

SWD1007E List memory allocation failure: status

Explanation: There was a memory allocation error, try using a larger region. Additional information may be given.

SWD1008W Invalid MVC day on line nn, skipping.

Explanation: The day component of an MVC date entry in the MVC work data set is invalid. The MVC is not included in further processing.

SWD1009W Invalid MVC year on line nn, skipping.

Explanation: The year component of an MVC date entry in the MVC work data set is invalid. The MVC is not included in further processing.

SSWD1013I MVCs selected: nn

Explanation: Issued at utility completion and indicates the number of MVCs in the specified Storage Class.

SWD1015I Completed with return code: rc

Explanation: Issued at utility completion.

SWD1016I Export VTVs selected for vaulting: nn

Explanation: The count of export VTVs successfully processed.

SWD1020W No MVCs selected

Explanation: Issued at utility completion when no MVCs were selected.

SWD1021W No VTVs selected

Issued at utility completion when no VTVs were selected.

SWD1050W Missing storage class for MVC mmmmmm on line nn, skipping

Explanation: No Storage Class name specified in the MVC report for an MVC. The MVC is skipped.

SWD1051W Missing volser on line nn, skipping.

Explanation: An MVC entry in the work MVC data set was missing a volser. The entry is ignored.

SWD1052W Invalid ACS location for MVC mmmmmmmm on line nn, skipping

Explanation: The MVCRPT for MVC mmmmmmmm had an ACS location that was not -- or 2 valid hexadecimal characters. As this is not an expected situation, the vault utility will skip this MVC. Check the MVCRPT listing in the SLSPRINT output to see if the reason that the MVC did not have a valid ACS location can be found. If so and the situation can be corrected, then the vault utility can be re-run. If not, then contact customer support.

SWD1100E Internal error, unknown TMS

Explanation: An internal error has occurred in the utility.

SWD1105E Unable to load SWSADMIN utility module.

Explanation: The VTCS SWSADMIN utility load module cannot be loaded. The VTCS load libraries must be either in the STEPLIB concatenation or the linklist.

SWD1106I Commencing SWDTMS01 run at ddddddd

Explanation: Prints the time/date at the start of a run. This is the time/date that will be written to the time stamp data set at the conclusion of a successful run.

SWD1107E Utility SWSADMIN report error rc=nn, run terminated.

Explanation: The VTCS SWSADMIN utility encountered an error with rc=*nn* during *report* phase processing. The SWDTMS01 utility terminates.

SWD1108E Unable to open SWSADMIN utility command file: [additional information]

Explanation: The VTCS SWSADMIN utility cannot open its command file referenced by the SLSIN DD. Additional information may be given.

SWD1109I Starting SWSADMIN utility for report

Explanation: Issued when starting the VTCS SWSADMIN utility for the named report phase.

SWD1110I Unable to open SWSADMIN utility report file: [additional information]

Explanation: Unable to open the SWSADMIN utility report file referenced by the SWDRPT DD. Additional information may be given.

SWD1111I Force option specified, selecting all MVCs

Explanation: The -F parameter was specified to force processing of all MVCs ignoring the contents of the time stamp data set. However, the time stamp data set will updated at the conclusion of a successful run.

SWD1112I Utility SWSADMIN completed report, rc=nn.

Explanation: The VTCS utility completed processing the report phase with rc=nn.

SWD1901E Missing storage class name

Explanation: No Storage Class name was specified after the –S parameter switch.

SWD1902E Storage class list too long

Explanation: The list of storage classes after the –S parameter exceeds 60 characters.

SWD1905E Illegal option -option

Explanation: An unknown parameter switch was given.

SWD1950W Unable to open the parameter dataset for read

Explanation: The utility could not read the parameter file data set.

SWD1951W Parameter dataset does not contain a valid parameter set

Explanation: The parameter file contains one or more invalid parameters.

SWD1952E Could not register parser module

Explanation: The program was unable to register the parser module.

SWD1955E Parse error rc - reason loc line nn: message

Explanation: There was a parameter error on or from the indicated line, the following *message* describes the problem. Occurs after message SWD1956E.

SWD1956E parameter_line

Explanation: The parameter line where the error starts or occurs on. Depending upon the error, it may be followed by a second line indicating the location of the error.

SWD1957E Error loading module name: message

Explanation: The utility could not load the module *name*, the following *message* describes the cause.

SWD1958W MVC mmmmm does not have any resident VTVs

Explanation: While processing the export manifest file, MVC mmmmmm was found to have no VTVs on it. This is only a warning and the MVC will still have TMS update statements generated to cause the MVC to be sent offsite.

SWD1959E Unable to append to command buffer: ssss

Explanation: There was a problem adding text to the internal command buffer used for executing a report. The string ssss indicates where in the code this occurred. This can usually be cause by application resource problems. Please contact your customer support if the problem persists.

SWD1960E Internal error, syntax definition doesn't match parameters

Explanation: While parsing the input parameters, the program could not match the data returned by the parser with it's own definition. This is an internal error and should be reported to StorageTek Support immediately.

SWDTMS02 Utility

The SWDTMS02 utility reads the output of an ExLM custom volume report and generates TMS update commands to allow returning MVCs from the DR vault and making these MVCs available for reuse.

Specifically, the SWDTMS02 utility:

- Creates the TMSUPDATE commands to modify TMS volume records for the exported MVCs. These commands set the required values to indicate the day the MVC will expire and can be returned to the onsite MVC pool.
- Creates the appropriate MVCMAINT commands to reset the Readonly bit to make these MVCs available for reuse after they're reentered into the ACS.
- For MVCs that contain VTVs with data sets that do **not** have retention periods, creates the MVCDRAIN commands to recall all current VTVs from the onsite copy of the MVC.

Parameters

None.

SWDTMS02 JCL Requirements

SWDPRM
: Optionally, specifies the parameter file used with the SWDTMS02 utility. The parameter file can be a PARMLIB member. For more information, see "VSM Vault Utilities Parameter Files" on page 166.

SWDEXLST
: Contains the output of an ExLM custom volume report that selects MVCs to be returned.

SWDTMS
: Contains the generated TMS commands for the selected MVCs.

SWDMVCDR
: Contains the generated VTCS MVCDRAIN commands for the selected MVCs.

SWDSIMP
: Contains MVCMAINT commands to set readonly off for all MVCs selected in SWDEXLST. Used for MVCs which contain 0 VTVs.

SWDDUP

Contains the generated MVCMAINT commands for all selected MVCs, where the returned MVCs are duplexed and therefore are marked as lost.

Caution: As described in "Returning MVCs from the Offsite Vault for Reuse" on page 31, the SWDSIMP commands are the **only** input to MVCMAINT when returning MVCs where the data sets have identifiable expiration dates.

As described in "Returning MVCs from the Offsite Vault for Reuse" on page 31, **both** the SWDSIMP commands and the SWDDUP commands are **successively input** to MVCMAINT when returning MVCs where the data sets **do not** have identifiable expiration dates.

ExLM JCL Example for SWDTMS02 - Selecting MVCs with Less Than a Specified Percent Used

.Figure 82 shows example JCL for ExLM to Select Vaulted MVCs with less than a specified percent used where the TMS specified is CA-1.

```
//EXLM3   JOB (ACCOUNT),'EXLM MVC > 0 SAMPLE'
//*
//*********************************************************************
//* PRODUCE REPORT FILE  INDICATING MVCS TO DRAIN
//*   FOR INPUT TO VBCTMS02
//*  SELECTS MVCS WHICH ARE LESS THAN A CERTAIN % FULL
//* CORRECT DATA SET NAMES AS REQUIRED.
//* PREALLOCATE THE EXLM LIST DATA SET
//* SET CORRECT % FULL IN THE REPORT STATEMENT
//* SET CORRECT VAULT CODE IN THE REPORT STATEMENT
//*********************************************************************
//*
//STEP1    EXEC PGM=LCMMAIN,PARM='PAGESIZE(45)',REGION=4M
//STEPLIB  DD  DSN=SYSX.LCM.R600.LCMLINK,DISP=SHR
//         DD  DSN=SYSX.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=SYSX.HSC.V600.SLSLINK,DISP=SHR
//*
//*   PUT IN YOUR TMC NAME
//*
//LCMTMSDB DD DSN=SYSX.CA1.TMC,DISP=SHR,DCB=BUFNO=255
//*
//*   PREALLOCATE EXLM LIST
//*
//VDREXLST DD  DSN=YOURHLQ.VSMDR.EXLMLIST,DISP=OLD
//LCMMSGS  DD SYSOUT=*
//SORTMSGS DD SYSOUT=*
//LCMLOG   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//LCMPARM  DD *
 OPTIONS
    CHECK         /* RUN WITH CHECK THE FIRST TIME */
    NOSYNC        /* NO NEED TO DO SYNCHRONIZATION */
    /* SYNCVTV */
    NONSWAPPABLE
    TITLE('VSM DR MVCS TO DRAIN        ')
   ;
 TMS
    CA1           /* CHANGE AS APPROPRIATE */
    NAME('MY TMS NAME')
   ;
 /* SET MVCINUSE LE XX CORRECTLY FOR YOUR MVCS */
 /* SET SCVDRX TO THE STORAGE CLASS NAME WHICH NEEDS */
 /*    DEFRAGGING                          */
 REPORT VOLUME
    DDNAME(VDREXLST)
    POSTACTION
    STYLE(DATA)
    WHEN (MVC AND
          NOT INLSM AND
          DSN MATCHES 'STKVSM.SCVDRX.**' AND
          LOCCODE EQ 'VDR1' AND
          MVCINUSE LE XX AND
           (EXPDT EQ 'PERM'D)
        /*(EXPDT EQ '1999365'D)   USE INSTEAD OF ABOVE FOR RMM */
         )
      COLUMN(SERIAL)
       ;
 MANAGE
    ACSID(00)   /* PUT IN CORRECT ACSID(S) */
   ;
/*
//
```

Figure 82. Example JCL for ExLM to Select Vaulted MVCs with Less Than a Specified Percent Used

ExLM JCL Example for SWDTMS02 - Selecting MVCs with No Current VTVs

Figure 83 shows example JCL for ExLM to select MVCs with no current VTVs.

```
//EXLM2   JOB (ACCOUNT),'EXLM MVC 0 SAMPLE'
//*
//*********************************************************************
//* PRODUCE REPORT FILE  INDICATING MVCS TO RETURN FROM OFFSITE
//*   FOR INPUT TO VBCTMS02
//*   SELECTS MVCS WHICH ARE 0% FULL
//*
//* CORRECT DATA SET NAMES AS REQUIRED.
//*
//* PREALLOCATE THE EXLM LIST DATA SET
//*
//* SET CORRECT VAULT CODE IN THE REPORT STATEMENT
//*********************************************************************
//STEP1    EXEC PGM=LCMMAIN,PARM='PAGESIZE(45)',REGION=4M
//STEPLIB  DD  DSN=SYSX.LCM.R600.LCMLINK,DISP=SHR
//         DD  DSN=SYSX.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=SYSX.HSC.V600.SLSLINK,DISP=SHR
//*
//*   PUT IN YOUR TMC NAME
//*
//LCMTMSDB DD DSN=SYSX.CA1.TMC,DISP=SHR,DCB=BUFNO=255
//*
//*   PREALLOCATE EXLM LIST
//*
//VDREXLST DD  DSN=YOURHLQ.VSMDR.EXLMLIST,DISP=OLD
//LCMMSGS  DD SYSOUT=*
//SORTMSGS DD SYSOUT=*
//LCMLOG   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//LCMPARM  DD *
 OPTIONS
    CHECK         /* RUN WITH CHECK THE FIRST TIME */
    NOSYNC        /* NO NEED TO DO SYNCHRONIZATION */
    /* SYNCVTV */
    NONSWAPPABLE
    TITLE('VSM DR MVC BACK FROM OFFSITE')
   ;
 TMS
    CA1           /* CHANGE AS APPROPRIATE */
    NAME('MY TMS NAME')
    ;

 REPORT VOLUME
    DDNAME(VDREXLST)
    POSTACTION
    STYLE(DATA)
    WHEN (MVC AND
            NOT INLSM AND
          DSN MATCHES 'STKVSM.**' AND
          LOCCODE EQ 'VDR1' AND
          MVCVTVCNT EQ 0 AND
           (EXPDT EQ 'PERM'D)
        /*(EXPDT EQ '1999365'D)   USE INSTEAD OF ABOVE FOR RMM */
          )
      COLUMN(SERIAL)
      ;

 MANAGE
    ACSID(00)   /* PUT IN CORRECT ACSID(S) */
    ;

/*
```

Figure 83. Example JCL for ExLM to Select Vaulted MVCs with No Current VTVs

JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with Less Than a Specified Percent Use

Figure 84 and Figure 85 on page 160 shows example JCL to run the SWDTMS02 and SWSADMIN utilities to process MVCs with less than a specified percent use.

```
//V790582T JOB ,'SWDTMS02',MSGCLASS=H,NOTIFY=V790582,
//         CLASS=A,REGION=2M
//*************************************************************
//* DELETE OUTPUT DATA SETS
//*************************************************************
//STEP1    EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//DD01     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.TMSCMDS
//DD02     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.SIMPLEX
//DD03     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.DUPLEX
//DD04     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.MVCDRAIN
//*******************************************************
//* RUN SWDTMS02 AGAINST FILE OUTPUT FROM EXLM
//*******************************************************
//STEP2    EXEC PGM=SWDTMS02
//STEPLIB  DD  DSN=PROD2.VSMDR.LOAD,DISP=SHR
//SWDPRM   DD  DSN=PREFIX.PARMLIB(PRMFILE2),DISP=SHR
//SWDEXLST DD  DSN=VSM02.VSMDR.EXLMLIST,DISP=SHR
//SWDTMS   DD  DSN=VSM02.VSMDR.TMSCMDS,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDSIMP  DD  DSN=VSM02.VSMDR.SIMPLEX,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDDUP   DD  DSN=VSM02.VSMDR.DUPLEX,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDMVCDR DD  DSN=VSM02.VSMDR.MVCDRAIN,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
```

Figure 84. Example JCL for the SWDTMS02 Utility

```
//*******************************************************************
//*
//* CONDITIONALLY RUN MAINT AGAINST THE DUPLEX CARDS TO SET LOST ON FOR
//* OFFSITE MVCS WITH LESS THAN A CERTAIN PERCENTAGE UTILIZATION
//*
//*******************************************************************
//STEP3    EXEC PGM=SWSADMIN,
//             COND=(4,LE,STEP2)
//STEPLIB  DD  DSN=PROD2.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=PROD2.HSC.V600.SLSLINK,DISP=SHR
//SLSCNTL  DD  DSN=PROD2.HSC.V600.DBASE1,DISP=SHR
//SLSCNTL2 DD  DSN=PROD2.HSC.V600.DBASE2,DISP=SHR
//SLSSTBY  DD  DSN=PROD2.HSC.V600.DBASE3,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SLSIN    DD  DSN=VSM02.VSMDR.DUPLEX,DISP=SHR
/*
//*******************************************************************
//*
//* CONDITIONALLY INPUT THE DRAIN CARDS TO DRAIN THE
//* OFFSITE MVCS WITH LESS THAN A CERTAIN PERCENTAGE UTILIZATION
//*
//*******************************************************************
//STEP4    EXEC PGM=SWSADMIN,
//             COND=(4,LE,STEP2)
//STEPLIB  DD  DSN=PROD2.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=PROD2.HSC.V600.SLSLINK,DISP=SHR
//SLSCNTL  DD  DSN=PROD2.HSC.V600.DBASE1,DISP=SHR
//SLSCNTL2 DD  DSN=PROD2.HSC.V600.DBASE2,DISP=SHR
//SLSSTBY  DD  DSN=PROD2.HSC.V600.DBASE3,DISP=SHR
```

Figure 85. Example JCL for the SWDADMIN Utility

JCL Example - Running SWDTMS02 and SWSADMIN to Process MVCs with No Current VTVs

Figure 86 and Figure 87 on page 162 shows example JCL to run the SWDTMS02 and SWSADMIN utilities to process MVCs with no current VTVs.

```
//V790582T JOB ,'SWDTMS02',MSGCLASS=H,NOTIFY=V790582,
//         CLASS=A,REGION=2M
//*************************************************************
//* DELETE OUTPUT DATA SETS
//*************************************************************
//STEP1    EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//DD01     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.TMSCMDS
//DD02     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.SIMPLEX
//DD03     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.DUPLEX
//DD04     DD  UNIT=SYSDA,DISP=(MOD,DELETE),SPACE=(TRK,0),
//             DSN=VSM02.VSMDR.MVCDRAIN
//*******************************************************
//* RUN SWDTMS02 AGAINST FILE OUTPUT FROM EXLM
//*******************************************************
//STEP2    EXEC PGM=SWDTMS02
//STEPLIB  DD  DSN=PROD2.VSMDR.LOAD,DISP=SHR
//SWDPRM   DD  DSN=PREFIX.PARMLIB(PRMFILE2),DISP=SHR
//SWDEXLST DD  DSN=VSM02.VSMDR.EXLMLIST,DISP=SHR
//SWDTMS   DD  DSN=VSM02.VSMDR.TMSCMDS,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDSIMP  DD  DSN=VSM02.VSMDR.SIMPLEX,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDDUP   DD  DSN=VSM02.VSMDR.DUPLEX,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SWDMVCDR DD  DSN=VSM02.VSMDR.MVCDRAIN,
//             UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(5,1),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
```

Figure 86. Example JCL for the SWDTMS02 Utility

```
//*******************************************************************
//*
//* CONDITIONALLY RUN TMSUPDTE TO UPDATE THE TMC RECORDS
//* FOR MVCS WITH 0 VTVS TO EXPIRE THEM
//*******************************************************************
//STEP3    EXEC PGM=TMSUPDTE,                      /* For CA-1
/*             PGM=CATINQR,                       /* For CA-TLMS
/*             CIMMUP,                            /* For CONTROL-M/Tape
/*             PGM=IKJEFT01,                      /* For DFSMSrmm
/*             ZARAUTL,                           /* For Zara
//         COND=(4,LE,STEP2)
//TMSRPT   DD   SYSOUT=*                          /* For CA-1
//SYSIN    DD   DSN=VSM02.VSMDR.TMSCMDS,DISP=SHR  /* Not for DFSMSrmm
//SYSTSIN  DD   DSN=VSM02.VSMDR.TMSCMDS,DISP=SHR  /* For DFSMSrmm
//SYSTSPRT DD   SYSOUT=*                          /* For DFSMSrmm
//*******************************************************************
//*
//* CONDITIONALLY RUN MVCMAINT TO SET READONLY OFF
//* FOR "ZERO VTVS" OFFSITE MVCS RETURNING TO THE DATA CENTER
//*
//*******************************************************************
//STEP4    EXEC PGM=SWSADMIN,
//             COND=(4,LE,STEP2)
//STEPLIB  DD  DSN=SYS2.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=SYS2.HSC.V600.SLSLINK,DISP=SHR
//SLSCNTL  DD  DSN=SYS2.HSC.V600.DBASE1,DISP=SHR
//SLSCNTL2 DD  DSN=SYS2.HSC.V600.DBASE2,DISP=SHR
//SLSSTBY  DD  DSN=SYS2.HSC.V600.DBASE3,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SLSIN    DD  DSN=VSM02.VSMDR.SIMPLEX,DISP=SHR
/*//*******************************************************************
//*
//* CONDITIONALLY INPUT THE DRAIN CARDS TO DRAIN THE
//* OFFSITE MVCS TO REMOVE STORAGE CLASS
//*
//*******************************************************************
//STEP4    EXEC PGM=SWSADMIN,
//             COND=(4,LE,STEP2)
//STEPLIB  DD  DSN=PROD2.VTCS.V600.SWSLINK,DISP=SHR
//         DD  DSN=PROD2.HSC.V600.SLSLINK,DISP=SHR
//SLSCNTL  DD  DSN=PROD2.HSC.V600.DBASE1,DISP=SHR
//SLSCNTL2 DD  DSN=PROD2.HSC.V600.DBASE2,DISP=SHR
//SLSSTBY  DD  DSN=PROD2.HSC.V600.DBASE3,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SLSIN    DD DSN=VSM02.VSMDR.MVCDRAIN,DISP=SHR
/*
```

Figure 87. Example JCL for the TMS update and SWDTMS02 Utilities

Sample Input and Output

Figure 88 and Figure 89 show, respectively, sample input to and output from the SWDTMS02 utility with CA1 as the TMS, as generated by the job streams shown in Figure 85 on page 160 and Figure 86 on page 161.

```
2002-2-22                        STORAGE TECHNOLOGY CORPORATION - ExLM 5.0.0          PAGE 1
07:37:21



                                 DR Volumes in Offsite to be Returned to ACS          VOLUME


VSM02.VSMDR.EXLMLIST


MVC039

MVC040
```

Figure 88. Sample SWDTMS02 Input

.

```
VSM02.VSMDR.TMSCMDS



VOL MVC039,NODSN,NOINTAL

VER DSN17=.MVCTAPE.RESERVED

REP EXPDT=2002/022

VOL MVC040,NODSN,NOINTAL

VER DSN17=.MVCTAPE.RESERVED

REP EXPDT=2002/022


VSM02.VSMDR.SIMPLEX


 MVCMAINT MVC(MVC039) READONLY(OFF)

 MVCMAINT MVC(MVC040) READONLY(OFF)
```

Figure 89. Sample SWDTMS02 Output

Return Codes

Table 21 describes return codes from the SWDTMS02 utility.

Table 22. SWDTMS02 Return Codes

Return Code	Description
0	Utility ran successfully and selected MVCs to process.
4	Utility ran successfully but did not select MVCs to process.
8	Utility abended.

Messages

SWD1950W Unable to open the parameter dataset for read

Explanation: The utility could not read the parameter file data set.

SWD1951W Parameter dataset does not contain a valid parameter set

Explanation: The parameter file contains one or more invalid parameters.

SWD1952E Could not register parser module

The program was unable to register the parser module.

SWD2000I Number of volumes read: nn

Explanation: The number of volumes read from the ExLM report.

SWD2001E Error opening output DD name dddddd [additional information]

Explanation: Error opening the output DD dddddd. Additional information may be provided.

SWD2003E Error whilst reading report in dddddd

Explanation: There was an error reading the ExLM report from DD dddddd.

SWD2004W DD dddddd could not be closed correctly. [additional information]

Explanation: There was an error when attempting to close the output DD dddddd. Additional information may be provided.

SWD2006E Error writing to DD dddddd [additional information]

Explanation: There was an error while writing to DD dddddd. Additional information may be provided.

SWD2007E Error opening input DD dddddd [additional information]

Explanation: Error opening the ExLM report input DD dddddd. Additional information may be provided.

SWD2008W Negative expiry period specified in parameter file: nn, using default.

Explanation: The EXPiRyPeRioD statement specified negative date *nn* so the default date (3 days) is used instead.

VSM Vault Utilities Parameter Files

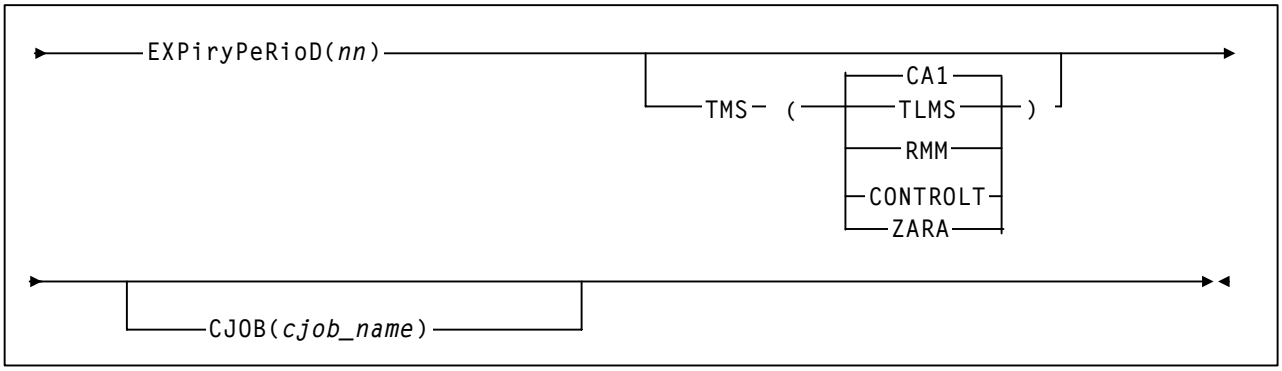
You can specify job options for the VSM Vault Utilities in a parameter file instead of entering command line arguments. The parameter file parameters are more powerful and flexible than the command line arguments. VSM Vault Utilities parameter files have the following requirements:

- Command line arguments override parameter file settings.
- Parameter file statements are free-form. That is, they can begin and end in any column are not limited to conventional 80-byte card input. All input columns are scanned. Text editor line numbers, which are traditionally used in columns 73 through 80, should **not** be used because they are interpreted by the parser as errors.
- A blank is the continuation character.
- You can specify control statements in lowercase, uppercase, or mixed case. The syntax of the parameter file statements shows valid abbreviations as uppercase.
- Write comments in any of the following styles:
  - /* Comment */ Anywhere spaces are allowed. This style can span multiple lines.
  - ..parms; // Anything from double slashes to end of line.
- Parameter files can be PARMLIB members.

EXPIRYPERIOD

The EXPiryPeRioD statement specifies the days until the MVC expires and, optionally, change the CJOB name, for the SWDIMS02 utility.

Syntax.



Parameters.

nn

The number of days until the MVC expires. Valid values are whole integers 0 and above. The default is 3.

TMS

The TMS where the SWDIMS02 utility sends the output.

CA1

CA-1.

TLMS

CA-TLMS.

CONTROLT

CONTROL-M/Tape

RMM

DFSMSrmm

ZARA

Zara

CJOB

Specifies the TMS attribute of tape data sets containing the name of the creation job. Used for TLMS. For more information, see Step 2 page 55.

cjobname

The CJOB name. There is no default value.

Example. Figure 90 shows an example of a SWDTMS02 utility parameter file that sets the MVC expiration period to one day and specifies CA-1 as the TMS.

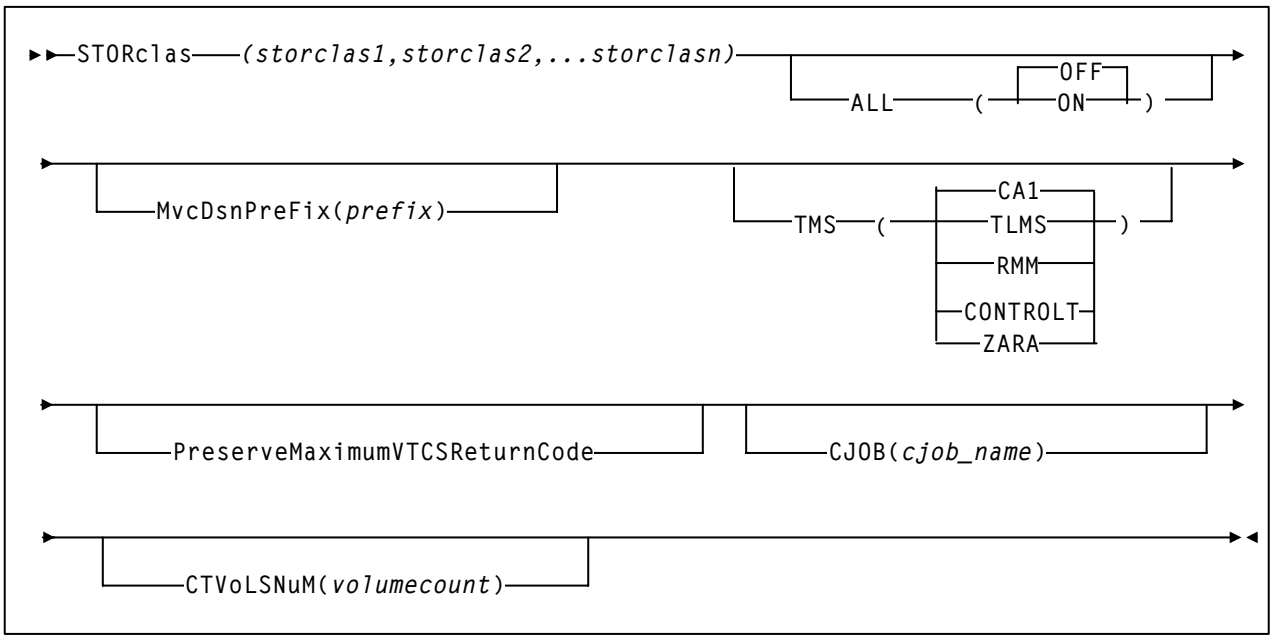
```
EXPIRYPERIOD(1) TMS(CA1)
```

Figure 90. EXPiryPeRioD ***Statement Example***

STORCLAS

The STORclas statement specifies the Storage Classes and other processing options for the SWDIMS01 utility. You must specify a Storage Class on the STORCLAS statement.

Syntax.



Parameters.

storclas1,storclas1,...storclasn

A comma-delimited list of Storage Classes for the SWDIMS01 utility to process. Each Storage Class name can be a maximum of 17 characters.

ALL

Specifies how the SWDIMS01 selects MVCs.

OFF

Select only MVCs that are resident in an ACS, which eliminates the attempts to vault volumes not found in the ACS (the default).

ON

Select MVCs even if they are not resident in an ACS. This select list can be used to determine which MVCs should be vaulted for comparison with the TMS systems.

MvcDsnPreFiX

The HLQ of the name of the data set that the SWDIMS01 utility assigns to the MVC in the TMC. For more information, see Step 7 on page 27.

prefix

The HLQ, which has a maximum length of 17 characters. The default, if not specified, is STKVSM.

TMS

The TMS for which the SWDIMS01 utility is creating update cards.

CA1

CA-1.

TLMS

CA-TLMS.

CONTROLT

Control-M/Tape

RMM

DFSMSrmm

ZARA

Zara

PreserveMaximumVTCSReturnCode

This parameter causes SWDIMS01 to modify the return code it issues. If specified, the maximum return code from the various VTCS calls made by the SWDIMS01 program is incorporated into the SWDIMS01 return code as follows.

SWDIMS01 return code = 100 * Maximum VTCS return code + original SWDIMS01 return code.

For example, if the VTCS EXPORT call made by SWDIMS01 was not able to select all MVCs, the return code for the EXPORT would be 4. If at least one MVC was selected by the export then the SWDIMS01 return code would be 0. If the parameter is specified, the SWDIMS01 return code would then be modified to become 400.

Use of this parameter allows the user to determine whether there were any problems encountered that may have caused some eligible MVCs to not be selected to be sent offsite. These MVCs can then be handled separately but it would ensure that the TMS update step is still run when at least one MVC is selected.

CJOB

Specifies the TMS attribute of tape data sets containing the name of the creation job. Used for TLMS. For more information, see Step 2 page 55.

cjobname

The CJOB name. There is no default value.

CTVOLSNUM

Specifies a value to be generated for the VOLSNUM keyword. The value is ignored unless the TMS parameter has a value of CONTROLT.

volumecount

An integer that will be specified on the VOLSNUM.

Examples. Figure 91 shows an example of a SWDIMS01 utility parameter file that does the following:

- Processes Storage Classes SCVDR1 and SCVDR2.
- Specifies CA-1 as the TMS.
- Sets the HLQ to VDR0302.
- Only selects MVCs that are ACS resident.
- Specifies that the return code currently produced by the SWDIMS01 program is modified to include the maximum return code received from the various VTCS calls made by the SWDIMS01 program.

```
STORCLAS(SCVDR1,SCVDR2) TMS(CA1) MVCDSNPREFIX(VDR0302) ALL(OFF) PMVTCSRC
```

Figure 91. STORCLAS ***Statement Example for CA-1***

Figure 92 shows an example of a SWDIMS01 utility parameter file that does the following:

- Processes Storage Classes SCVDR1 and SCVDR2.
- Specifies Control-M/Tape as the TMS.
- Sets the HLQ to VDR0302.
- Only selects MVCs that are ACS resident.
- Specifies that the return code currently produced by the SWDIMS01 program is modified to include the maximum return code received from the various VTCS calls made by the SWDIMS01 program.
- Specifies 10 as the value to match the value specified for the VOLSNUM keyword of the Control-M/Tape DSNADD command.

```
STORCLAS(SCVDR1,SCVDR2) TMS(CCONTROLT) MVCDSNPREFIX(VDR0302) ALL(OFF)
PMVTCSRC CTVOLSNUM(10)
```

Figure 92. STORCLAS ***Statement Example for Control-M/Tape***

A Sample Offsite Vaulting Work Flow

In “Creating the Export MVCs and Vaulting Them Offsite” on page 29 and “Returning MVCs from the Offsite Vault for Reuse” on page 31, we show these parts of the process as separate procedures with separate JCL jobs. However, most shops will elect to have a **single** daily job stream ending in the VMS run and ExLM eject run. This section graphically shows a sample workflow that represents a single daily job stream. **Note that** order is important to produce the desired effects.

In this sample flow, we first identify offsite fragmented MVCs and defragment them. This allows the targets of the defragmentation process to be sent offsite today, and the offsite defragmented MVCs to be brought back onsite today. After you understand this sample workflow plus the basic procedures provided in “Migration, Export, Offsite Vault, and MVC Reuse” on page 23, you can vary this workflow to suit your site’s needs.

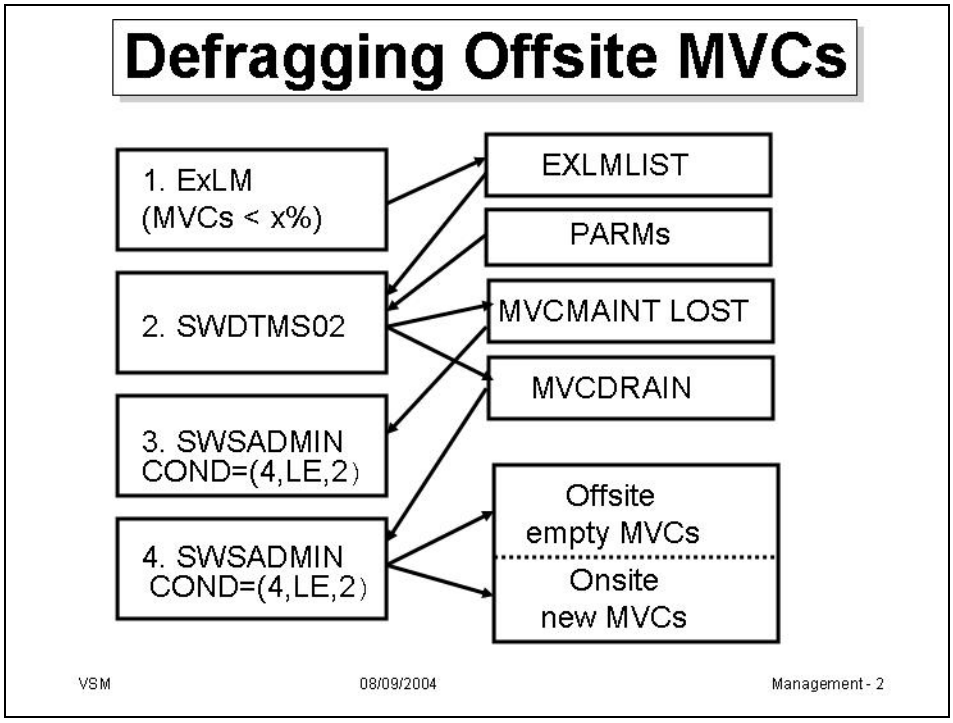


Figure 93. ***Defragmenting Offsite MVCs***

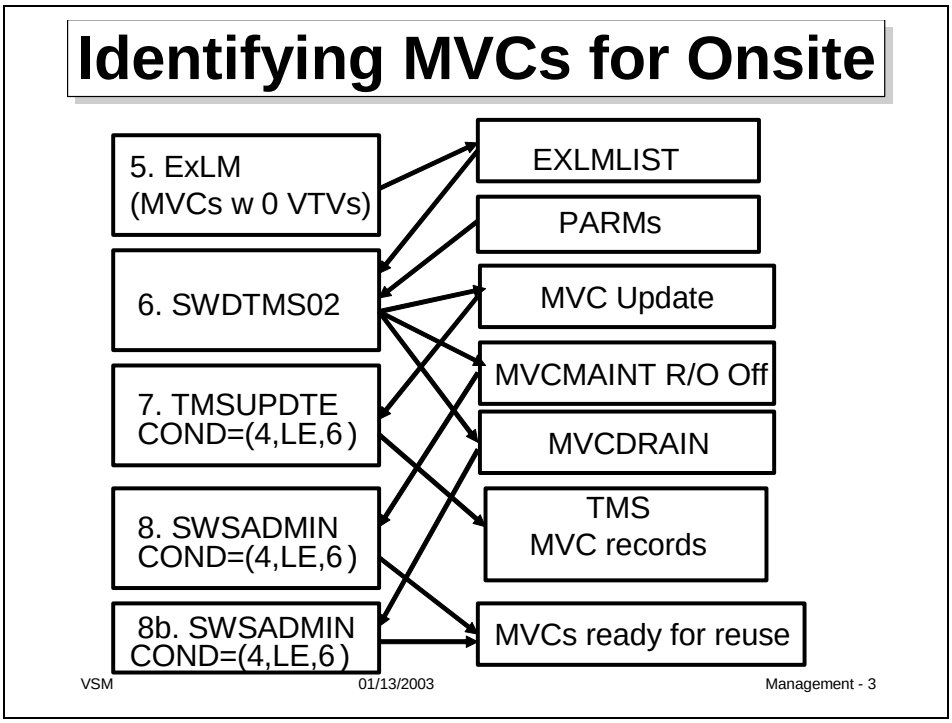


Figure 94. Identifying MVCs to Return Onsite

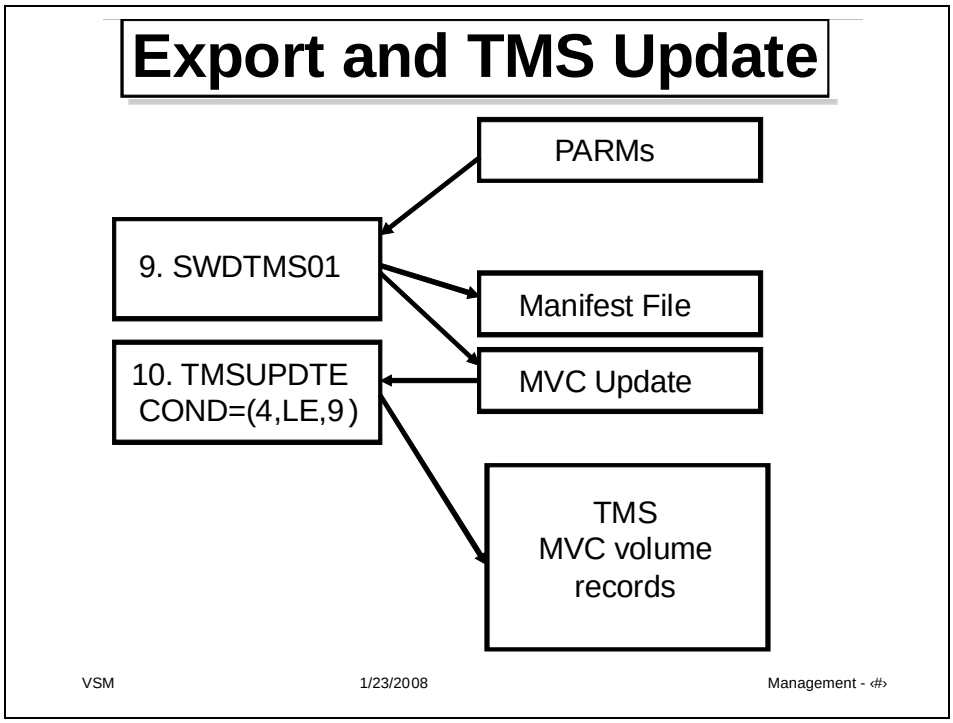


Figure 95. Exporting MVCs and Updating the TMS

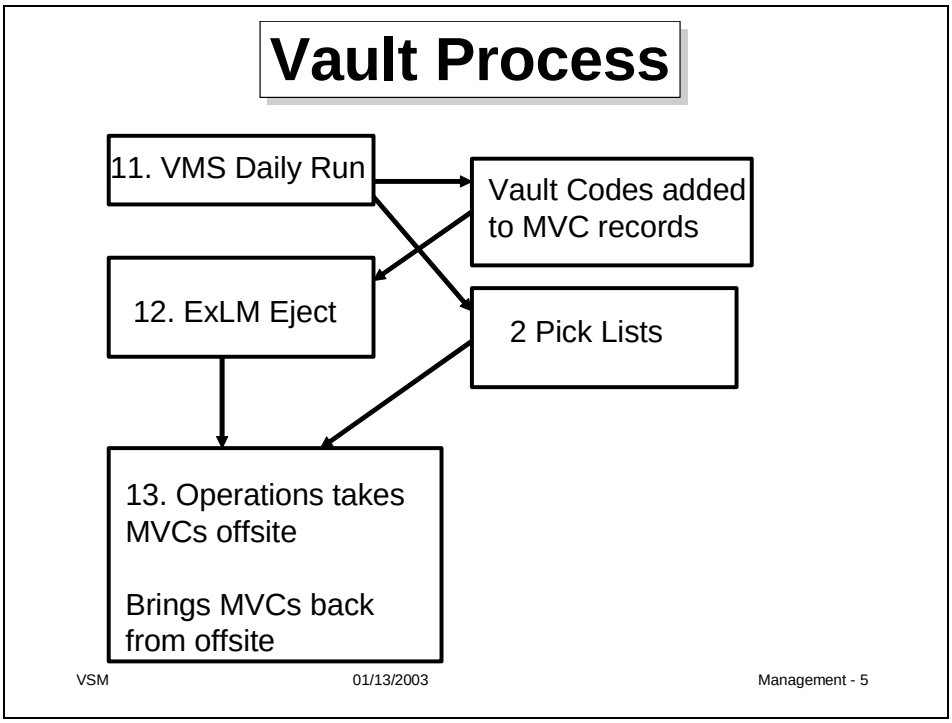


Figure 96. Vaulting Offsite MVCs

Offsite Vaulting Configuration Record

Your Sites Values - Data Sets with Retention Periods

Table 23. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Data Sets with Retention Periods - Your Site's Values

Management Class (`mgmt-class-name`)	**Retention Period** **RETPD(LT,**`n`**)**	**Storage Class** (`stor-clas-name`)	**Named MVC Pool** (`poolname`)	**TAPEREQ Data Set Mask** (`mask`)

Your Sites Values - Catalog Controlled Data Sets

Table 24. Disaster Recovery Management Classes, Storage Classes, and Named MVC Pools - Catalog Controlled Data Sets without Retention Periods - Your Sites Values

Management Class	Storage Classes	Use	Named MVC Pool	TAPEREQ Data Set Mask

APPENDIX **A**

Additional Information

Oracle Corporation (Oracle) offers several methods for you to obtain additional information.

Oracle's External Web Site

Oracle's external Web site provides marketing, product, event, corporate, and service information. The external Web site is accessible to anyone with a Web browser and an Internet connection. The URL for the Oracle external Web site is: http://www.oracle.com/us/index.html

The URL for Oracle/StorageTek/Sun storage information for is: http://www.oracle.com/us/products/servers-storage/storage/index.html

Sun/StorageTek Documentation

The URL for Sun/StorageTek documentation is:

http://docs.sun.com/app/docs

Oracle Global Partners

The Oracle Global Partners site provides information about solutions available with Oracle's partners:

http://www.oracle.com/us/partnerships/index.html

Third-Party Web Sites

Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Oracle’s Global Offices

You may contact any of Oracle’s worldwide offices to discuss complete storage, service, and support solutions for your organization. You can find contact information at:
http://www.oracle.com/corporate/contact/global.html

Customer Support

For more information about Oracle support (including for Sun/StorageTek branded products) see:
http://www.oracle.com/us/support/index.htm?origref=
http://www.oracle.com/us/products/index.html

Conventions for Reader Usability

Conventions are used to shorten and clarify explanations and examples within this book.

Typographic

The following typographical conventions are used in this book:

- **Bold** is used to introduce new or unfamiliar terminology.
- Letter Gothic is used to indicate command names, filenames, and literal output by the computer.
- Letter Gothic Bold is used to indicate literal input to the computer.
- *Letter Gothic Italic* is used to indicate that you must substitute the actual value for a command parameter. In the following example, you would substitute your name for the "username" parameter.
- Logon *username*
- A bar (|) is used to separate alternative parameter values. In the example shown below either username or systemname must be entered.
- Logon *username*|*systemname*
- Brackets [ ] are used to indicate that a command parameter is optional.
- Ellipses (...) are used to indicate that a command may be repeated multiple times.
- The use of mixed upper and lower case characters (for non–case sensitive commands) indicates that lower case letters may be omitted to form abbreviations. For example, you may simply enter **Q** when executing the **Quit** command.

Keys

Single keystrokes are represented by double brackets [[ ]] surrounding the key name. For example, press [[ESC]] indicates that you should press only the escape key.

Combined keystrokes use double brackets and the plus sign (+). The double brackets surround the key names and the plus sign is used to add the second keystroke. For example, press [[AL]] + [[C]] indicates that you should press the alternate key and the C key simultaneously.

Enter Command

The instruction to "press the [[ENTER]] key" is omitted from most examples, definitions, and explanations in this book.

For example, if the instructions asked you to "enter" **Logon pat**, you would type in **Logon pat** *and* press [[ENTER]].

However, if the instructions asked you to "type" **Logon pat**, you would type in **Logon pat** and you would *not* press [[ENTER]].

Warnings, Cautions, and Notes - Software

The following are used in software documentation.

Caution – Information necessary to keep you from corrupting your data.

Tip – Information that can be used to shorten or simplify your task or they may simply be used as a reminder.

Note – Information that may be of special interest to you. Notes are also used to point out exceptions to rules or procedures.

Warnings, Cautions, and Notes - Hardware

The following are used in hardware documentation.

Note – A note provides additional information that is of special interest. A note might point out exceptions to rules or procedures. A note usually, but not always, follows the information to which it pertains.

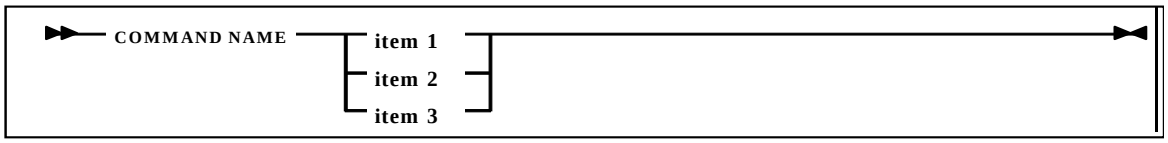
Caution – A caution informs you of conditions that might result in damage to hardware, corruption of data, or corruption of application software. A caution always precedes the information to which it pertains.

Warning – Possible Physical Injury. A warning alerts you to conditions that might result in long-term health problems, injury, or death. A warning always precedes the information to which it pertains.

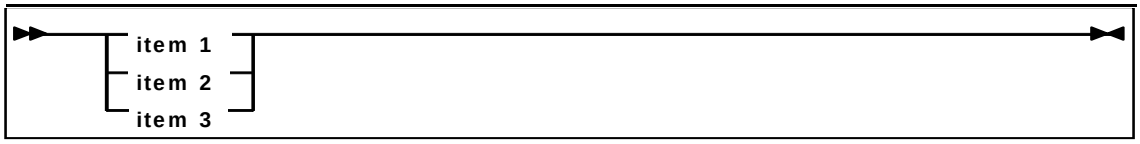
Syntax

Syntax flow diagram conventions include the following:

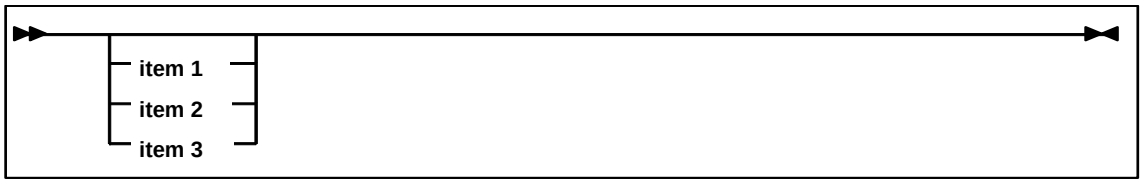
Flow Lines—Syntax diagrams consist of a horizontal baseline, horizontal and vertical branch lines and the command text. Diagrams are read left to right and top to bottom. Arrows show flow and direction.



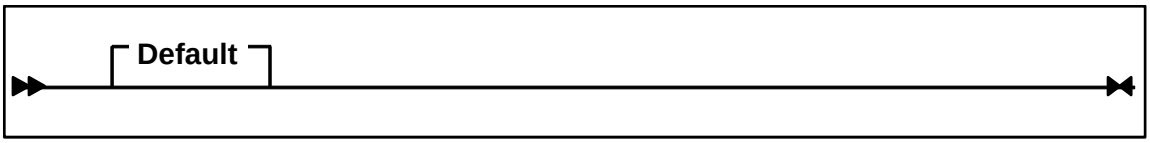
Single Required Choice—Branch lines (without repeat arrows) indicate that a single choice must be made. If one of the items to choose from is on the baseline of the diagram, one item must be selected.



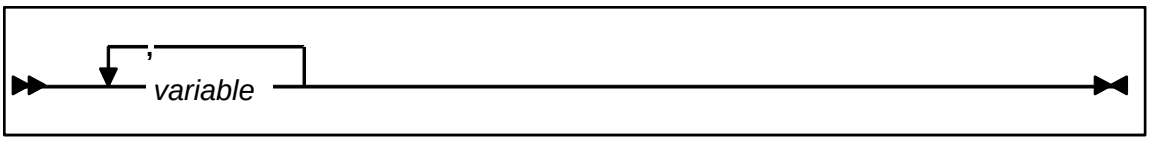
Single Optional Choice—If the first item is on the line below the baseline, one item may optionally be selected.



Defaults—Default values and parameters appear above the baseline.



Repeat Symbol—A repeat symbol indicates that more than one choice can be made or that a single choice can be made more than once. The repeat symbol shown in the following example indicates that a comma is required as the repeat separator.



Keywords—All command keywords are shown in all upper case or in mixed case. When commands are not case sensitive, mixed case implies that the lowercase letters may be omitted to form an abbreviation.

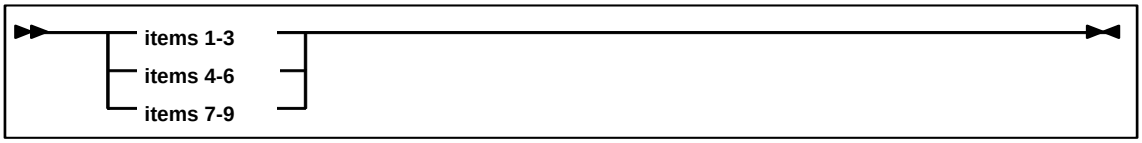
Variables—Italic type is used to indicate a variable.

Alternatives—A bar (|) is used to separate alternative parameter values.

Optional—Brackets [ ] are used to indicate that a command parameter is optional.

Delimiters—If a comma (,), a semicolon (;), or other delimiter is shown with an element of the syntax diagram, it must be entered as part of the statement or command.

Ranges—An inclusive range is indicated by a pair of elements of the same length and data type, joined by a dash. The first element must be strictly less than the second element.



Lists—A list consists of one or more elements. If more than one element is specified, the elements must be separated by a comma or a blank and the entire line must be enclosed by parentheses.