StorageTek Enterprise Library Software

Disaster Recovery and Offsite Data Management Guide

Version 7.2

Revision 05
Part Number: E37155-05

Submit comments about this document to STP_FEEDBACK_US@ORACLE.COM.

Disaster Recovery and Offsite Data Management Guide

E37155-05

Preface

Oracle's StorageTek™ Enterprise Library Software (ELS) is a solution consisting of the following base software:

- StorageTek™ Storage Management Component (SMC)
- StorageTek™ Host Software Component (HSC)
- StorageTek™ Virtual Tape Control Software (VTCS)
- StorageTek™ Concurrent Disaster Recovery Test (CDRT)

Additionally, the following software is provided with the ELS package:

- StorageTek™ Library Content Manager (LCM) (formerly ExLM). LCM includes an enhanced version of the product formerly known as Offsite Vault Feature.
- StorageTek™ Client System Component for MVS Environments (MVS/CSC)
- StorageTek™ LibraryStation

Audience

This guide is for StorageTek or customer personnel who are responsible for planning and implementing Disaster Recovery (DR) and offsite data management solutions.

Prerequisites

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

- Z/OS operating system
- JES2 or JES3
- Enterprise Library Software (ELS)

About This Book

ELS Disaster Recovery and Offsite Data Management Guide contains the following:

- "Introduction to Disaster Recovery" on page 1, which is an introduction to and overview of Disaster Recovery solutions.
- For offsite data management, see:
  - "Doing Physical Exports and Imports" on page 21 is all about using the EXPORT and IMPORT facilities. EXPORT creates portable MVCs at the source site, which you then eject from the source ACS, physically transport them to another site, and use IMPORT to bring them into the target site.
  - "Using the ELS External Vaulting Feature" on page 29 describes using this DR feature...which, in turn, uses EXPORT and IMPORT plus additional commands and utilities to automate the process of vaulting MVCs offsite.
  - "Using Clustered VTSS Configurations" on page 75 provides everything you wanted to know (and then some) about implementing Clustered VTSSs, including Extended Clustering (3 plus VTSSs per cluster) that is new for ELS 7.0 and above. By itself, Clustered VTSS is not, strictly speaking, a DR solution. Because of the redundancy/backup/distributed capabilities of Clustered VTSS, however, it easily lends itself to DR solutions. For example, I set up Management and Storage Classes so that I make two MVC copies, one at a local site, the other at a remote site. For more information, also see "Clustered VTSS Examples" on page 157.
- For specific DR solutions, see:
  - "Using Cross-TapePlex Replication in a DR Solution" on page 49, which describes how to use the Cross-Tapeplex Replication Feature to set up a complete disaster recovery solution for tape data
  - "Using the Concurrent Disaster Recovery Test Software" on page 93, which tells how to use this feature to validate your site's DR solutions. **Note that** for ELS 7.2, CDRT supports Virtual Library Extension (VLE) 1.0and above.
  - "Creating System Recovery Points in VSM Environments" on page 141, which tells how to implement DR recovery points in a VSM environment.

  Collectively, all of these chapters discuss methods of moving data from one location to another and managing the data. These processes can be used with other processes for DR/business continuance/business resumption. The ELS Vault feature lets you manage offsite DR and Long Term Retention (LTR) volumes and also manual rack ("floor") volumes.

  So you have a full range of DR and offsite data management solutions...how do you select the best option for your site from these offerings?

How Do I Choose a Solution for My Needs?

The solutions described in *ELS Disaster Recovery and Offsite Data Management Guide* vary in cost and accompanying benefits, where the general rule of thumb is that higher cost solutions allow faster recovery times or shorter data management cycles. For example, the export/import of MVCs (optionally, using the ELS External Vault Feature) is relatively low cost, but recovery takes time simply because of the need to move MVCs from one site to another.

Cross-TapePlex Replication (CTR), on the other hand, is relatively expensive. You need two sites, multiple VTSSs, RTDs and MVCs at both sites, and so forth. CTR uses VTSS Clustering to copy data over FICON or ESCON *cluster link*s directly from a VTSS in one TapePlex to a VTSS in another TapePlex. The data movement (and the metadata movement) is electronic between sites, so backup and recovery potentially occurs much faster than with External Vault.

What's New in This Guide?

Revision 05

Revision 05 of this guide contains information about remote library support described in "Configuring a Remote Library" on page 71

Revision 04

Revision 04 of this guide contains updates to the CDRT facility described in "Using the Concurrent Disaster Recovery Test Software" on page 97.

Revision 03

Revision 03 of this guide contains technical updates and corrections.

Revision 02

Revision 02 of this guide contains technical updates and corrections, including information about the enhancements described in TABLE P-1.

TABLE P-1 Updates to DR Guide 7.2, Revision 02

This Enhancement...	...is described in...
Support for VSM 6	■ "Clustering with TCP/IP Connections" on page 94 ■ "VSM5 to VSM 6 Cluster with TCP/IP CLINKs, Cross-Connected VLEs" on page 180 ■ "VSM 6 to VSM 6 "Tapeless" Cluster with TCP/IP CLINKs" on page 183

Contents

CHAPTER 1

Introduction to Disaster Recovery

Best Practices for Enterprise Disaster Recovery (DR) basically consist of designing and implementing fault-tolerant hardware and software systems that can survive a disaster ("business continuance") and resume normal operations ("business resumption"), with minimal intervention and, ideally, with no data loss. Building fault-tolerant environments to satisfy Enterprise DR objectives and real-world budget constraints can be expensive and time consuming and requires a strong commitment by the business.

DR plans typically address one or more of these types of disasters:

- Extensive or extended IT facility damage due to natural disaster (earthquake, storm, flooding, and so forth) or other causes (fire, vandalism, theft, and so forth).
- Extended loss of IT facility critical services, e.g., loss of power, cooling or network access.
- Loss of key personnel.

The DR planning process begins by identifying and characterizing the types of disasters a business must survive and resume operations. The planning process identifies high-level business continuance (BC) and business resumption (BR) requirements, including the required degree of fault tolerance. The product of DR planning is a recovery and resumption architecture for fault-tolerant systems, applications and data to support these requirements subject to established constraints. Typical DR constraints include recovery time objective (RTO), recovery point objective (RPO) and available budget. The DR architecture plus the business constraints leads to DR procedures that integrate all system elements in a true "end-to-end" fashion to guarantee predictable results for the overall DR process.

Fault tolerant systems typically achieve robustness and resiliency through **redundancy**. Often achieved at great expense, a fully redundant system has no single point of failure within its architecture and can operate during and resume operations from the worst possible disaster within its limits. Space shuttle and aircraft flight control systems are good examples of fully-redundant systems. Less critical IT applications typically use less robust systems with lower redundancy. These systems are less costly to construct and will necessarily incur a service outage after disaster strikes, during which time the business works to reinstate its recoverable systems, applications and data.

Ultimately, the nature of a business, its customer requirements, and the available budget for DR are key factors in formulating DR requirements. A comprehensive DR solution can cost a lot...but it has to be architected. You can't just throw money, hardware, and software at a potential disaster and hope to survive and resume your business operations. If you plan and architect intelligently, on the other hand, you may have to incur longer outages, degraded service, or both until full services can resume, but you can still have a dependable, limited DR solution.

Understand, however, that perhaps **no** amount of planning can anticipate and respond to **all** possible DR scenarios. For example, what begins as an apparently trivial problem on one system can spread over time to affect other systems in different ways, all adding up to a disaster for which there is no recovery scenario. Similarly, a business's ability to honor service agreements may suffer if key assumptions do not hold true...for example, if key parts or service are unavailable, or if the DR provider's delivery capability is not as robust as advertised. The real key, however is that if a disaster occurs that **exceeds** the worst-case scenario that you planned for, recovery may not be possible.

Defining the Recovery Time Objective (RTO)

RTO is a service level objective of the time it takes to achieve a desired operational capability after a disaster has occurred. For example, business requirements may dictate the RTO that all production systems are up and running at 80% of pre-disaster capability within 30 minutes of any unplanned outage that would last longer than one hour (if no DR capability existed). RPO processing time, availability of qualified IT staff and the complexity of manual IT processes required after a disaster are examples of constraints that may shape RTO determination. RTO does **not** apply to fully fault tolerant systems because these systems recover implicitly during and after a disaster, with no service interruption.

DR planners might set different RTOs for some or all of the defined BC requirements. Different types of business operations may require different RTOs, for example different RTOs for online systems versus batch windows. Further, different RTOs may apply in stages for a phased DR plan, where each phase has a defined RTO. Even further, a recoverable application may have different RTOs for each of its various service levels.

BC data availability requirements are extremely important to RTO planning. When data that must be input to the DR recovery process is not present at the disaster recovery site, the time it takes to retrieve the data onsite will delay RTO. For example, data residing in offsite storage vaults will take time to retrieve. Recovery can proceed quickly if up-to-date input data is duplicated at the recovery site before the start of disaster recovery operations.

Defining the Recovery Point Objective (RPO)

RPO is a business continuance objective that dictates the business state, or business currency, that is achieved after the disaster recovery process has reinstated all recoverable systems. Conceptually, RPO is a known “rollback” or synchronization target state prior to disaster. That is, RPO is the post-disaster recovery point from which interrupted recoverable applications can resume processing. All transactions occurring during the interval between the RPO and the time of disaster are unrecoverable. RPO does **not** apply to fully fault tolerant systems because disaster does not affect the business continuance of these systems.

FIGURE 1-1 illustrates RPO concepts by suggesting various recovery points for DR planners to consider. Planning must ensure the desired RPO is feasible vis-à-vis the chosen RTO and vice versa. Generally, disaster recovery plans that mandate RPOs nearer to the time of disaster require greater fault tolerance and are more costly to implement versus more distant RPOs. As with RTO, DR planners might set different RPOs for different BC requirements, DR plan phases or application service levels.

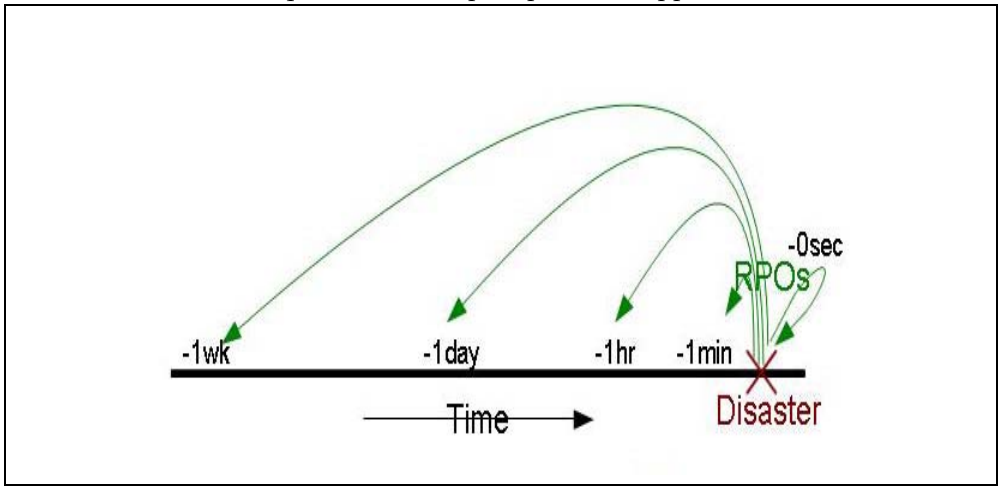

FIGURE 1-1 Recovery Point Objectives

More broadly, RPO planning must identify all supporting elements that must be present to reinstate each recoverable system, including data, metadata, applications, platforms, facilities and personnel. Planning must also ensure these elements are available at the desired level of business currency for recovery. BC data currency requirements are especially crucial to RPO planning. For example, if BC requirements dictate a one-hour RPO, any data or metadata that feeds the recovery process must be current up to the RPO, otherwise the RPO cannot be achieved. The organization's DR processes will specify the procedures to achieve all defined RPOs within the stated RTOs.

System metadata required for RPO recovery includes OS catalog structures and tape management system information. These items must be updated during the disaster recovery process to enable all of the chosen RPOs. For example, to ensure consistency among the various metadata inputs to the DR recovery process, existing data sets that will be recreated at RPO must be uncataloged; data sets updated between the RPO and the time of disaster must be restored to the version that existed at or before the RPO; and any tape-related catalog changes must be synchronized with the tape management system.

Handling Temporary Outages

Disaster recovery provides remediation for very long-term outages that would render a production site unusable for an extended period of time. While the remainder of this introduction addresses disaster recovery practices, it might be just as important to develop procedures to mitigate relatively brief outages that could negatively affect production if left unchecked. Consider, for example, a service outage where certain hardware or network facilities are unavailable for an hour or two, but production can continue during this outage in “degraded mode” with a few quick, temporary adjustments. A temporary outage procedure would document how to isolate the problem, what changes to make, whom to notify and how to revert to the normal operating environment after service is restored.

Key Concept: Synchronization Point Recovery

Restarting production applications at defined RPOs is a key activity performed during true disaster recovery and during DR testing. The most resilient DR environments will ensure every recoverable application, whether sourced or developed in-house, enforces a core DR requirement: namely, that the application is designed to restart from a planned epoch, called a synchronization point, in order to mitigate the effects of an unscheduled interruption during its execution. When an interrupted application is restarted at a synchronization point, the results are the same as if the application had not been interrupted.

The restart procedure for a recoverable application depends on the nature of the application and its inputs. Quite often, the application restart procedure for true disaster recovery or DR testing is the same procedure used for restarting the application should it fail during a normal production run. Where possible, reusing production restart procedures for true disaster recovery or DR testing simplifies the creation and maintenance of DR procedures and leverages these proven procedures.

In the simplest case, a recoverable application is a single job step with only one synchronization point, which is merely the beginning of the program invoked by that step. In this case, the recovery procedure might be as simple as resubmitting the interrupted job. A slightly more complex restart procedure might involve uncataloging all of the output data sets produced by the application during its last run, and then restarting the application.

The restart procedures for applications having multiple internal synchronization points to choose from might not be so simple. Applications that use checkpoint/restart techniques to implement these synchronization points periodically record their progress and can, for example, use recorded checkpoint information to restart at the last internal synchronization point recorded prior to an interruption. Restart procedures will conform to the requirements of each synchronization point. When checkpointing is in use, data sets associated with a checkpoint must not be expired, uncataloged or scratched while the checkpoint remains valid for application recovery.

An easy way to establish a synchronization point for a job step that modifies its existing input data sets is to make a backup copy of each modifiable data set before running the step. These modifiable input data sets are easily identified by searching for JCL attribute DISP=MOD in DD statements or in dynamic allocation requests. In the event of a job step failure or interruption, simply discard any modified input data sets, restore those input data sets from backup copies, and restart the step from the restored copies. These backup copies are also useful to restart a failed or interrupted job step that had expired, uncataloged or scratched the originals.

Relating RPO to Synchronization Point Recovery

When the RPO aligns with a synchronization point, performing the application restart procedure that was developed for this synchronization point will resume the application from this origin as if no interruption has occurred (FIGURE 1-2). All transactions processed after this RPO up to the disaster are presumed unrecoverable.

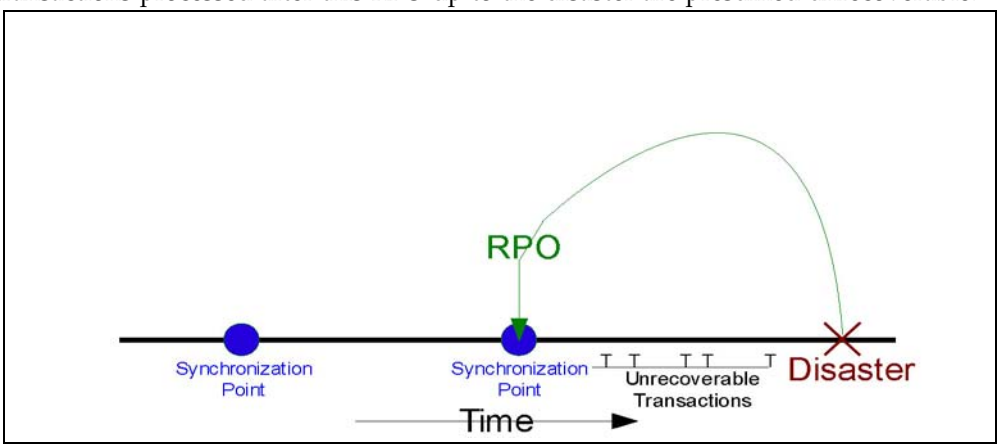

FIGURE 1-2 RPO at Synchronization Point

At other times, BC requirements may justify placing the RPO between synchronization points. In these cases, inter-synchronization point recovery relies on supplemental data describing any critical application state changes or events occurring after the most recent synchronization point is established.

Consider, for example, the RPO of one minute prior to disaster. Suppose a recoverable application is designed to use checkpoints to record its progress, but suppose the overhead of taking these checkpoints at one-minute intervals is intolerable. One solution is to take checkpoints less frequently and to log all transactions committed between checkpoints. This transaction log then becomes supplemental input data used by the checkpoint recovery process to restart from an RPO beyond the most recent synchronization point. In this example, the application restart procedure accesses the most recent checkpoint data and applies the supplemental transaction log to reinstate all committed transactions processed after the checkpoint and prior to the RPO (FIGURE 1-3) In this way, synchronization point recovery can achieve a target RPO using input data from multiple sources. All transactions processed after the RPO up to the disaster are presumed unrecoverable.

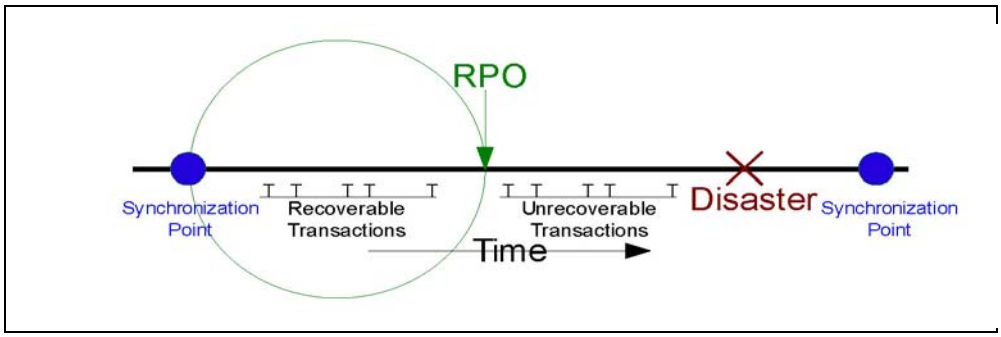

FIGURE 1-3 RPO Between Synchronization Points

Planning for Data High Availability (D-HA)

Data is often one of the most precious assets held by a business. Many companies take great care and make extra investment to safeguard business-critical data against loss, and to ensure data is available for its intended purpose when needed. A firm that cannot cope with critical data loss might well suffer from disastrous consequences.

Perhaps the most common way to protect against data loss is by storing copies of critical data on different storage media or subsystems, and by storing some of these copies at different physical locations. Copies stored on removable storage media, including magnetic cartridge tape, CD-ROM and DVD are typically vaulted at offsite storage locations. Extra copies are also typically stored onsite at IT facilities where applications can process that data.

Creating and storing critical data copies increases data redundancy and improves data fault tolerance. For removable media, and in particular for magnetic cartridge tape, increasing data redundancy alone is usually not enough to ensure data is also highly available to the applications that will use it. For example, Oracle's VSM system for mainframe virtual tape stores data on physical tape volumes called MVCs. VSM is capable of making MVC copies automatically to improve data redundancy and to reduce risk due to a media failure or a misplaced tape cartridge. A production VSM system utilizes many specialized hardware components to retrieve data stored on an MVC, including a VTSS buffer device, an automated tape library and library-attached tape drives called RTDs, which are also attached to the VTSS buffer device. Host applications depend on all of these VSM components operating together to retrieve data from MVCs. Even though most people would not regard a single component failure as a disaster on par with losing an entire data center in an earthquake, it certainly might become impossible to retrieve any MVC data if a single VSM critical component fails without a backup, no matter how many redundant MVC copies exist. Thus, while creating MVC copies is a proven best practice to mitigate vulnerability and risk, it does not always sufficiently guarantee data high availability (D-HA) in the presence of faults.

D-HA requirements are key business continuance requirements for DR planning. D-HA is typically achieved by increasing redundancies to eliminate single points of failure that would prevent applications from accessing data amidst storage system faults. For example, a VSM system that includes redundant components improves VSM system fault tolerance. Installing multiple VTSS devices, redundant SL8500 handbots and multiple RTDs aims to eliminate VSM single points of failure along the data path from the application to the critical data stored on an MVC. The VSM architecture is designed throughout to support adding redundant components to increase fault tolerance and promote D-HA.

Highly-Available Physical Tape

Oracle's mainframe tape automation solutions enable D-HA for physical tape applications by storing redundant copies of data in different ACSs within a tapeplex, i.e., within a tape complex mapped by a single CDS. For example, applications running at an IT facility with a single tapeplex can easily store duplicate copies of tape data sets in one or more ACSs within that tapeplex. This technique improves D-HA by adding redundant media, tape transports and automated tape libraries.

In a simple case, an application stores redundant copies of a critical data set on two different cartridge tapes in a single SL8500 library with redundant electronics, dual handbots on each rail and two or more library-attached tape transports on each rail that are compatible with the data set media. To remove the SL8500 library as a potential single point of failure, a second SL8500 is added to the ACS to store even more redundant copies of the critical data set. To eliminate the IT facility itself as a single point of failure, redundant data set copies can be vaulted offsite or created at a remote ACS with channel-extended tape transports (FIGURE 1-4).

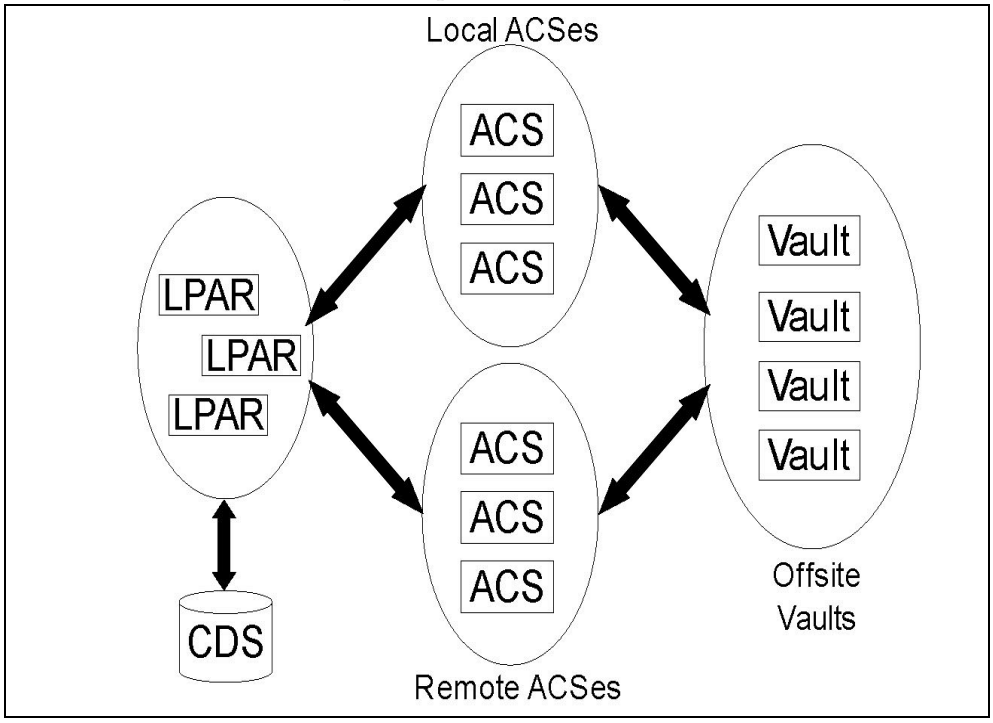

FIGURE 1-4 FD-HA Physical Tape Configuration

You can also make two or more copies of physical tapes in different physical locations when each location has its own independent CDS, i.e., when the hardware at each location represents a separate tapeplex. By using the SMC Client/Server feature and defining policies that direct data set copies to a remote tapeplex, jobs can create tape copies in an ACS in another tapeplex with no JCL changes.

Highly-Available Virtual Tape

VSM provides MVC N-plexing and clustering technologies to enable D-HA for mainframe virtual tape. VSM N-plexing involves creating multiple MVC copies (e.g., duplex, quadplex) in one or more ACSs for greater redundancy (FIGURE 1-5.). ACSs receiving N-plexed copies can be local libraries or remote ACSs with channel-extended tape transports. VSM migration policies control the movement of VTSS buffer-resident VTVs onto local or remote MVCs, which may be cycled to offsite vaults.

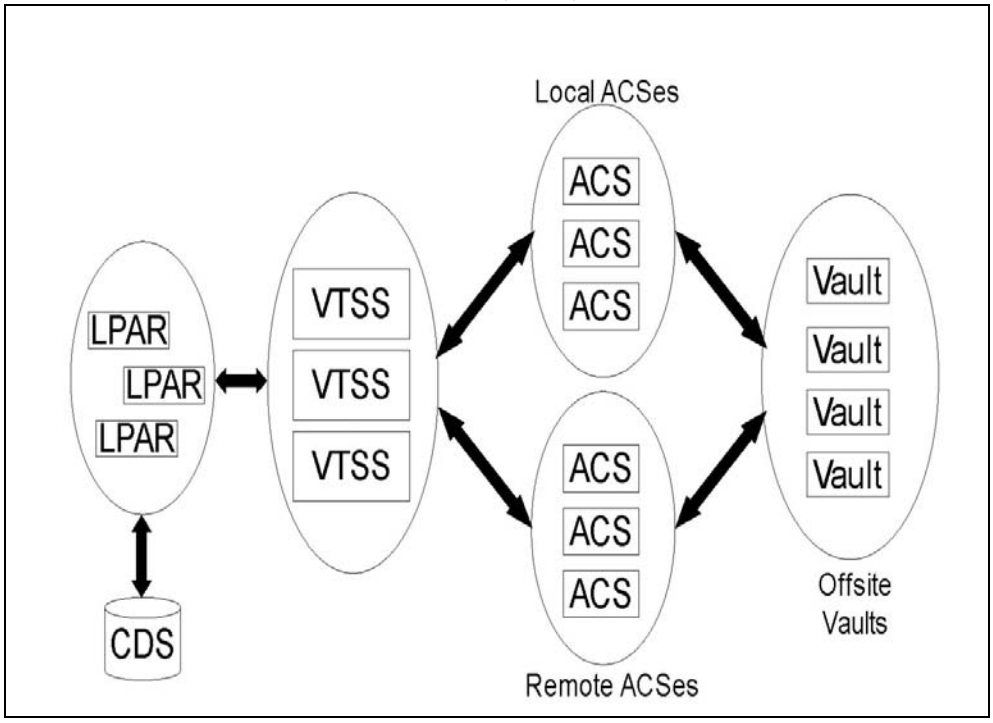

FIGURE 1-5 D-HA VSM N-plexing Configuration

A VSM cluster comprises two or more VTSS devices (nodes) networked for data interchange over a communications link (CLINK.) CLINKs are either unidirectional or bidirectional channels. The simplest VSM cluster configuration consists of two VTSS nodes in the same tapeplex linked with a unidirectional CLINK, but bi-directional CLINKs are commonly deployed (FIGURE 1-6). Each cluster node may be located at a different site. VSM uni-directional storage policies control the automatic replication of virtual tape volumes (VTVs) from VTSS A to VTSS B over a unidirectional CLINK. Bidirectional storage policies and bi-directional CLINKs enable VTSS A to replicate to VTSS B and vice versa.

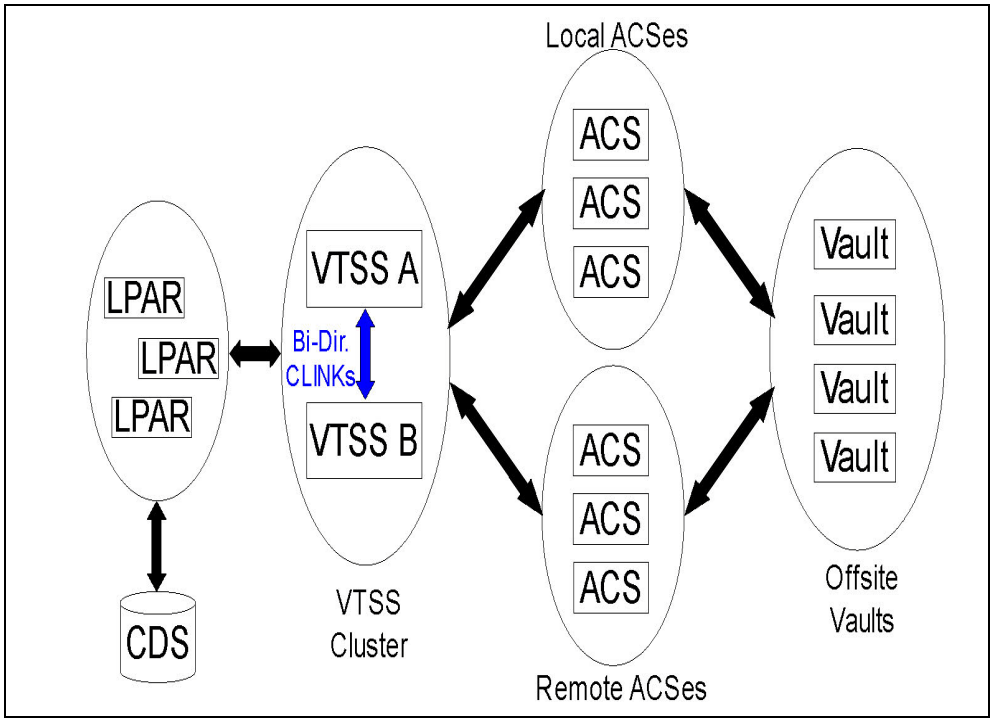

FIGURE 1-6 D-HA VSM Cluster Configuration

VSM Extended Clustering enables many-to-many connectivity among three or more VTSS devices in a tapeplex for even higher degrees of data availability (FIGURE 1-7). Installing VTSS cluster devices at two or more sites within a tapeplex as shown increases redundancy by eliminating each site as a single point of failure.

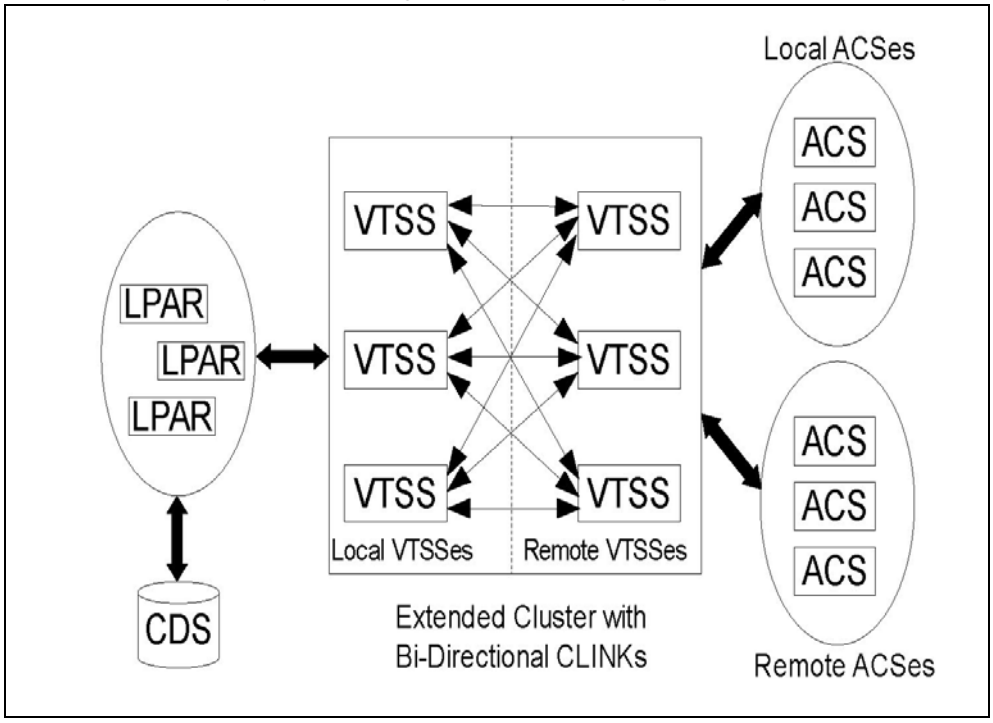

FIGURE 1-7 D-HA VSM Extended Cluster Configuration (offsite vaults not shown)

Oracle's LCM product streamlines the offsite vaulting processes for MVC volumes by managing the recycling process between vaults and production libraries. The LCM vaulting function schedules the return of vaulted MVC volumes when the amount of expired data exceeds a specified threshold.

A VSM Cross-Tapeplex Replication cluster (CTR cluster) allows VTSS cluster devices to reside in different tapeplexes and provides the capability to replicate VTVs from one tapeplex to one or more other tapeplexes, enabling many-to-many cluster replication models over uni-directional or bi-directional CLINKs (FIGURE 1-8.). Sending and receiving tapeplexes may be located at a different sites. Replicated VTVs are entered into the CDS for the receiving tapeplex as read-only volumes. This provides strong data protection against alteration by applications running in the receiving tapeplex. The CDS for the receiving tapeplex also indicates the CTR-replicated VTV copies are owned by the sending tapeplex and, as added protection, CTR ensures a tapeplex cannot modify any VTV it does not own.

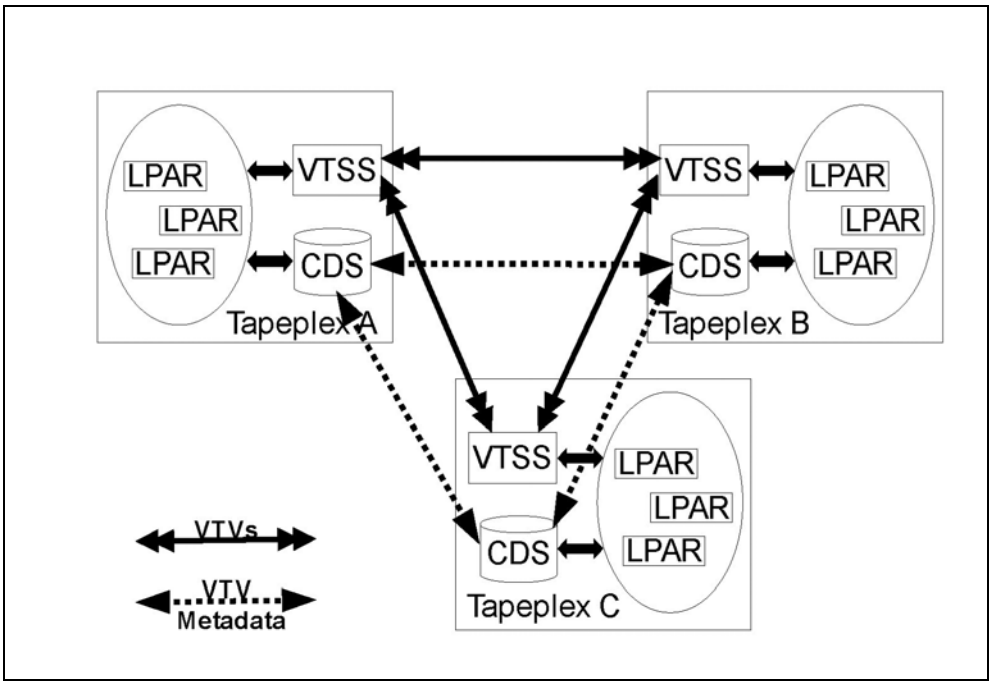

FIGURE 1-8 D-HA VSM Cross-Tapeplex Replication Configuration

D-HA and Synchronization Point Recovery

Creating multiple copies of physical volumes (MVCs or non-MVCs) improves data redundancy but these copies pose special considerations for synchronization point recovery. The most important aspect of synchronization point recovery is ensuring the data created at a synchronization point stays in a read-only state while it remains valid for disaster recovery usage. This means physical tape volume copies that could be used for disaster recovery must be kept in a read-only state. One way to do this is to send these copies to an offsite vault location where no tape processing capability exists. Be aware that any unprotected copies that undergo alteration are unusable for synchronization point recovery because the updated contents no longer reflect the associated synchronization point.

Virtual tape environments add an extra dimension to managing multiple volume copies for synchronization point recovery. VTV copies can exist in multiple VSM buffers and on multiple MVCs all at the same time. Even when all MVCs for a given VTV are vaulted offsite, VTV copies remaining onsite in VSM buffers can be modified. An updated buffer-resident VTV copy must not be used for synchronization point recovery unless this VTV belongs to a new synchronization point that invalidates the offsite copies vaulted for disaster recovery usage.

Conducting True Disaster Recovery

The success of a true disaster recovery operation rests on having an adequate DR site, trained personnel, a proven DR procedure, a recoverable production workload with synchronization points to meet defined RPO(s), and all input data and system metadata necessary to attain these RPOs. Input data and system metadata must be accessible at the DR site when needed and must be available at the required level(s) of currency. With careful planning, thorough preparation and well-rehearsed execution, true disaster recovery operations can flow smoothly according to plan to achieve the defined RPO(s) and RTO(s.)

Production data generated at the DR site must be adequately protected while the DR site is functioning as a production site. Suppose, for example, the D-HA architecture requires the production workload to replicate redundant data copies at three remote sites, and suppose the DR site is one of these remote replication sites prior to disaster. When the production site experiences a disaster and its workload is moved to the DR site, the DR site can no longer serve as a remote replication site for the production workload now running locally at that site. In order to meet the D-HA requirement of three remote replication sites, a new third remote replication site must be brought online for as long as production remains at the DR site. This example illustrates how a thorough analysis of D-HA requirements will enable DR planners to address all critical D-HA requirements that must be met when production is moved to a DR site.

A comprehensive DR plan encompasses not only the activities to reinstate production at the DR site but also includes the process to vacate the DR site when the production site is repaired and ready for business, assuming the DR site is only a temporary substitute for production. For example, when the production site is ready to resume operation, production data must be reinstated at that site. Methods include bi-directional clustering between the DR site and the production site, allowing enough time for production work running at the DR site to repopulate the former production site by data replication. It may, however, be necessary, or more timely or efficient, to simply transport physical MVCs back to the reinstated production site. The chosen methods will depend on the post-disaster recovery requirements.

Planning for DR Testing

True disaster recovery readiness is assessed by testing the efficiency and effectiveness of DR systems and procedures at recovering a production workload at a designated DR test site. The DR test environment may be a dedicated DR test platform, but usually it is more economical to share resources between production and DR test systems. DR testing performed in parallel with production and using resources shared with production is called concurrent DR testing. If an application must execute in parallel on production and DR test systems, DR planners should ensure these two instances of the application will not interfere with each another while they are running concurrently. Isolating the production and DR test systems on separate LPARs and limiting access to production data from the DR test system usually provides sufficient separation.

DR testing is often conducted piecemeal to allow targeted testing of different applications at different times, rather than testing recovery of the entire production environment all at once. Targeted testing is key to reducing the amount of dedicated hardware required for the DR test system. For example, if DR testing for a recoverable application requires only a small subset of VSM resources, those resources can be shared between the production and DR test systems and reassigned to the DR test system for the DR test cycle. This approach reduces DR test system hardware expense at the risk of impacting production system performance while the DR test is running. Typically, however, a DR test cycle dedicates only a small percentage of shared resources to the DR test system, and the diminished production environment is not greatly impacted by parallel DR testing. Nevertheless, some organizations have policies against altering or impacting production to facilitate DR testing.

Your auditors might require an exact match between DR test results and production results in order to certify the DR recovery process. One way to meet this requirement is to establish a synchronization point just ahead of a scheduled production run, save a copy of the production results, recover the production run at this synchronization point on the DR test site and compare the output against the saved production results. Any difference between results highlights a gap that must be investigated. Failure to address gaps in a timely fashion could put an organization's true disaster recovery capability at risk. No matter whether a DR test is designed to recover a complex workload or a single application, the DR test process must be performed using the same procedures that would be used for true disaster recovery. This is the only sure way to demonstrate the DR test was successful.

Data Movement for DR Testing

There are two methods to stage application data for DR testing at a DR test site: physical data movement and electronic data movement. Physical data movement involves transporting physical tape cartridges to the DR test site in a process described below called Physical Export/Import. Electronic data movement uses remote tape drives, remote RTDs or VSM cluster techniques to create copies of application data at a DR test site. Both of these data movement methods enable DR testing, but electronic data movement avoids physical data transfer and any potential problems with lost tapes, and so forth Electronic transfer also improves time to access data by placing it where needed for true disaster recovery, or by staging data in a VSM buffer ahead of a DR test cycle. Electronic data movement for virtual volumes can be done within a single tapeplex by using VSM Extended Clustering, or between two tapeplexes by using Cross-Tapeplex Replication. For data within a single tapeplex, Oracle's Concurrent Disaster Recovery Test (CDRT) software streamlines DR testing.

DR Testing With Physical Export/Import

Suppose you wish to perform DR testing for a production application that uses virtual and physical tape. Your goal is to test this application at the DR test site by repeating a recent production run and verifying the test output matches the recent production output. In preparation you'll need to save copies of any input data sets used by the production run and a copy of the production output for comparison.

Suppose the DR test site is isolated and shares no equipment with production. You could conduct the DR test using this Physical Export/Import process.

Production Site:

1. Make a copy of the requisite VTVs and physical volumes
2. Export those VTV copies
3. Eject associated MVC copies and physical volume copies from the production ACS
4. Transport ejected MVCs and physical volumes to the DR test site

DR Test Site:

1. Enter transported volumes into the DR ACS
2. Synchronize OS catalogs and tape management system with the entered volumes
3. Import VTV/MVC data
4. Run application
5. Compare results
6. Eject all volumes entered for this test
7. Transport ejected volumes back to production site

Production Site:

1. Enter transported volumes back into the production ACS.

This process allows DR testing to proceed safely in parallel with production since the DR test system is isolated from the production system. The DR test system has its own CDS, and the DR test process as above enters volume information into the DR test CDS in preparation for the DR test. This enables the recovered application to test with the same volumes and data set names it uses in production.

For virtual tape data sets, Oracle's LCM software vaulting feature simplifies placement of VTVs onto MVCs and streamlines the round-trip steps above to export and eject volumes at the production site, import those volumes at the DR test site and eject those volumes for movement back to the production site.

Physical Export/Import incurs site expenses for physical tape handling and courier expenses to transport tape cartridges between the production and DR test sites. Sensitive data moved by courier should be transported on encrypted tape cartridges. DR testing timeliness is affected by the time spent transporting and handling the tape cartridges moved between sites.

DR Testing With CDRT

With planning and with sufficient hardware at the production and DR sites, CDRT combined with electronic data movement can eliminate the need to transport physical tape cartridges to the DR site and enables concurrent DR testing more economically than maintaining an isolated, dedicated DR test site. CDRT enables DR testing of almost any production workload, configuration, RPOs or RTOs imaginable. The DR test procedure will include a few extra steps to start CDRT and clean up after a DR test.

Before running a DR test with CDRT, you should electronically move all of the application data and system metadata (OS catalog information and tape management system information) needed for the test to the DR test site. You can move application data electronically with VSM clustering or by migrating VTV copies to MVCs at the DR site. You then use CDRT to create a special CDS for the DR test system that mirrors the production CDS. The production and DR test systems are separate environments, and the DR test environment will use the special DR test CDS instead of the production CDS. Because CDRT creates the DR test CDS from information in the production CDS, it contains metadata for all of the volumes that were electronically moved to the DR test site prior to the DR test. This enables DR test applications to use the same volume serial numbers and tape data set names used in production.

CDRT enforces operational restrictions on the DR test system to prevent the DR environment from interfering with the production environment. You can strengthen these protections by using ELS VOLPARM/POOLPARM capabilities to define separate volser ranges for MVCs and scratch VTVs for exclusive use by CDRT. CDRT allows the DR test system to read from production MVCs and write to its own dedicated pool of MVCs that is logically erased at the end of each DR test cycle.

For virtual tape applications, CDRT requires at least one dedicated VTSS device for the duration of the DR test cycle. These dedicated VTSSs can be temporarily reassigned from production to facilitate a DR test, and the DR test VSM system may access production ACSs in parallel with the production workload.

FIGURE 1-9 on page 19 and FIGURE 1-10 on page 19 illustrate splitting a production VSM cluster to loan a cluster device to the CDRT DR test system, in this case VTSS2 at the DR test site. When this cluster is split, you must alter production policies to substitute migration for replication so VTSS1 will create redundant VTV copies at the DR site in ACS01, and so VTSS1 will not fill to capacity while the cluster is split. VTSS2 is taken offline to production and brought online to the DR test LPAR. In Fig. 9b, CDRT has created the DR test CDS from a remote copy of the production CDS. Only the production system may access volumes in VTSS1 and ACS00 throughout the DR test cycle, and only the DR test system may access VTSS2. The production and DR test systems share concurrent access to volumes in ACS01.

FIGURE 1-9 on page 19 and FIGURE 1-10 on page 19 maintaining a remote copy of the production CDS at the DR test site, for example by remote mirroring, to ensure an up-to-date production CDS is available at the DR site for true disaster recovery use. Note, however, that the DR test CDS created by CDRT from the remote CDS copy is a special DR test version of the production CDS only for use by CDRT.

Before re-forming the production cluster after the DR test cycle has ended, the DR VTSS must be purged to avoid production data loss, as would happen if VTSS2 contained a newer version of a VTV that also existed in VTSS1. You must also alter production policies to revert from migration to replication when the cluster is re-formed.

If splitting a production cluster as shown here isn't an option, an alternative is to maintain a separate VTSS at the DR site exclusively for DR testing. In this case, VTVs needed for the test will be recalled from MVC copies.

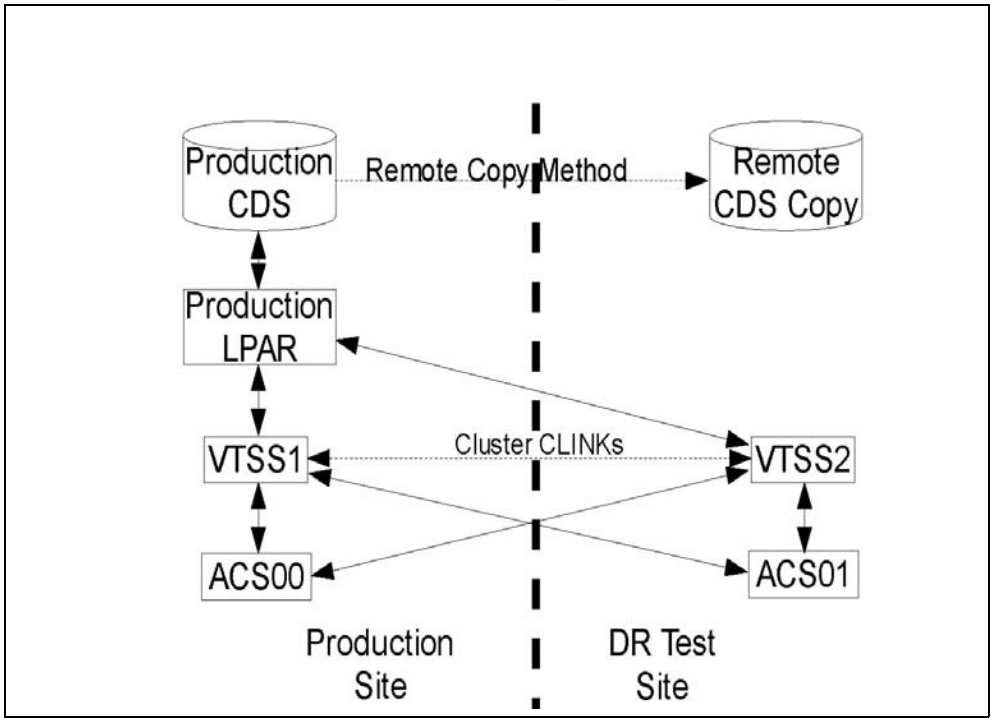

FIGURE 1-9 Production Cluster with Remote Cluster Node VTSS2 at DR Test Site

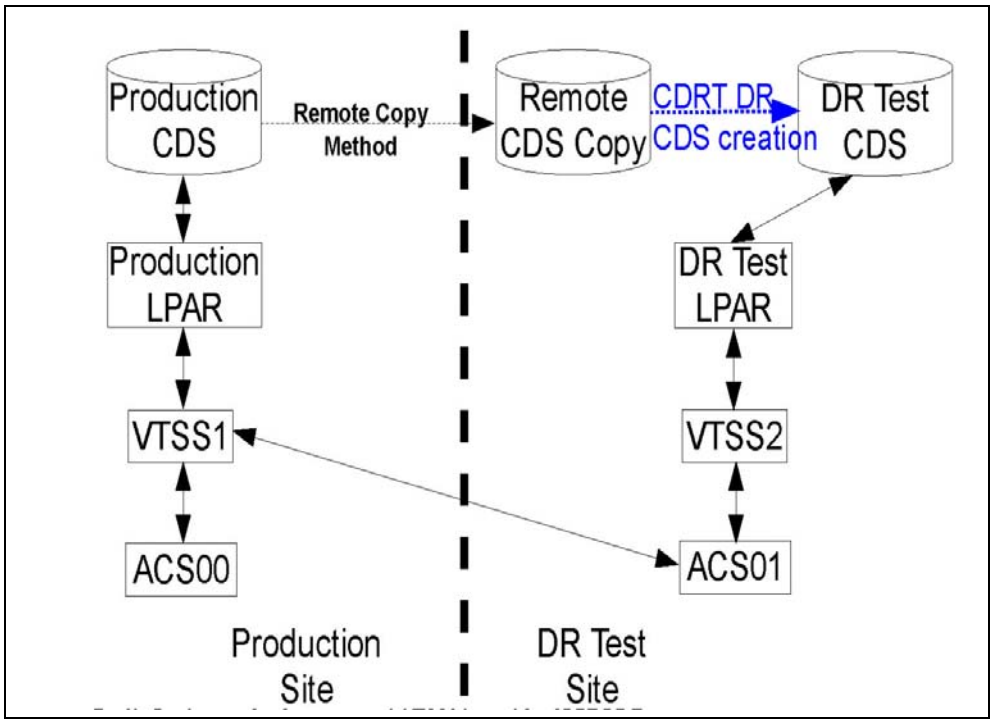

FIGURE 1-10 Production Configuration with VTSS2 loaned for CDRT DR Testing

DR Testing with VSM Cross-Tape Replication

VSM Cross-Tapeplex Replication enables symmetric, clustered, production tapeplex designs that facilitate DR testing without using CDRT, without requiring dedicated VTSS hardware purely for DR testing and without altering the production environment for DR testing. For example, CTR enables each production tapeplex to replicate data to the other production tapeplexes in the same CTR cluster. Production CTR peer-to-peer clusters can eliminate the need for a dedicated DR test site. CTR enables many different types of clustered tapeplex designs and facilitates DR testing of any production workload or configuration, with any feasible RPOs or RTOs.

In a simple example, a bi-directional CTR cluster joins two production tapeplexes symmetrically, and each tapeplex replicates data to the other TapePlex (FIGURE 1-11). A receiving tapeplex enters a replicated VTV into its CDS in read-only status and marks the VTV as owned by the sending tapeplex. In this example, DR testing for a tapeplex A application involves replicating application data at tapeplex B and recovering the application on tapeplex B.

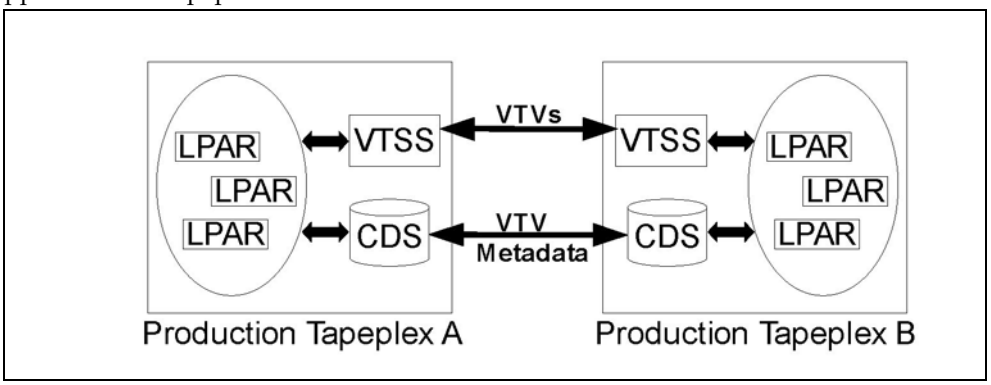

FIGURE 1-11 Symmetric Production CTR Cluster for DR Testing

The symmetry of this peered CTR cluster design means the recovered application being tested at the peer site runs exactly the same during a DR test as during production. The peer CDS contains all the replicated volume information needed for DR testing, which proceeds in parallel with production, and the same VTSS hardware supports concurrent use by production and DR test workloads. Production VTSS clusters may exist within each TapePlex and need not be split to share hardware across tapeplexes for DR testing. The production tapeplex on which application DR testing is performed cannot modify any CTR-replicated VTVs, so all replicated production data is fully protected during the DR test cycle. Most importantly, CTR-based DR testing guarantees a validated DR test procedure will deliver identical results during true disaster recovery.

SMC host software will issue a message if an attempt is made to update a CTR-replicated VTV, which serves to identify the application as one that modifies an existing input data set. Following best practices for managing synchronization points as above, you should ensure the production environment saves a copy of this data set before the application modifies it, should a backup copy be needed for synchronization point recovery.

CHAPTER 2

Doing Physical Exports and Imports

The EXPORT and IMPORT facilities give you the tools to create physically portable MVCs. At the source site, you use EXPORT to consolidate VTVs on MVCs (if necessary), and produce a Manifest File that describes the MVC contents (VTVs on the MVC). You then eject the MVCs from the source site, physically transport them to the target site, then IMPORT them, using the Manifest File to update the CDS with information on the imported MVCs and VTVs. **Note that** you can import VTVs into a CDS without VTCS being active. You then enter the MVCs into the target site.

Note –

- If you decide to return exported MVCs back to the source system, no special VTCS processing is required; just enter the MVCs into an LSM at the source system.
- For each VTV imported, the only MVC copies that will be created will be for MVCs that have been exported and imported via the same statements. This has a particular significance when importing Duplexed VTVs. Such a VTV will only have copies on both MVCs after the import if both MVCs are present in the same Manifest file and imported as a result of the same IMPORT statement.

You export by either of the following general methods:

- **Export by VTV or Management class**, which consolidates the selected VTVs onto a new set of MVCs. As Consolidation takes time and requires VTSS resources the preferred option is to perform Export by MVC or Storage Class. For more information, see "Exporting and Importing by Management Class" on page 22.
- **Export by MVC or Storage Class**. An export by storage class or MVC does not require any Consolidation post-processing of VTVs, and there is no data movement required. The export simply creates a Manifest File that describes the contents of the selected MVCs For more information, see "Exporting and Importing by Storage Class" on page 26.

Note – If you export by:

- **VTV volsers** - Use a TMS, LCM, or VTVRPT report to identify the required VTVs.
- **MVCs volsers** - Use an LCM or MVCRPT report to identify the required MVCs.
- **Management Class** - Review your Management Class definitions to identify the required Management Classes.
- **Storage Classes** - Review your Storage Class definitions to identify the required Storage Classes.

Exporting and Importing by Management Class

Example: Exporting by Management Class from the Source VSM System

This is the "send" phase of export/import, where we get the desired data packaged up and moved out of the source VSM system.

To export from a source VSM system, do the following:

1. **Identify the Management Classes used for export.**
2. **Export by Management Class:**

```
//EXPORT  EXEC PGM=SLUADMIN,PARM='MIXED' REGION=6M
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//MOVE1       DD DSN=hlq.REMOTE2,DISP=(,CATLG,DELETE),
//           UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
//SLSPRINT  DD SYSOUT=*
//SLSIN       DD *
 EXPORT MGMT (PAY,ACCOUNT) MANIFEST(MOVE1)
```

FIGURE 2-1 EXPORT by Management Class

In FIGURE 2-1, the output manifest file is MOVE1, which is required for the import. Because we exported by Management Class, EXPORT consolidates (makes copies of) the selected VTVs on *export MVCs*. The export MVCs are marked as read-only and as exported in the CDS, and are now available for ejection from a source system LSM.

These consolidated VTV copies are additional copies and are not recorded in the CDS. For example, if the VTV was duplexed before the export, the CDS records both duplexed copies, but the third additional copy used for consolidation is *not* recorded in the CDS. The original VTVs, therefore, are still available to the source system. You can use the data on the original VTVs or scratch and reuse them.

Caution – Schedule the export for a time when the exported data is not being updated!

---

3. **Remove the MVCs for export from the MVC pool.**

   For more information, see *Managing HSC and VTCS*.

4. **Eject the MVCs for export from a source VSM system LSM.**

   For more information, see *Managing HSC and VTCS.*

5. **If desired, at the source system, scratch or make unavailable the exported VTVs or reuse the data they contain.**

   After the export, the source system retains the CDS records of the exported VTVs and MVCs. The export MVCs are marked as exported and readonly in the source system CDS. At this point, you have two choices, depending on why you exported the VTVs:

   - **If you exported the VTVs to provide a backup copy at a second site**, leave the VTVs as readonly in the source system CDS so they cannot be updated.
   - **If you are permanently moving the exported VTVs to a second site**, then scratch or otherwise make them unavailable in the source system CDS. Use the HSC scratch utilities or the LCM SYNCVTV function to scratch exported VTVs.

Example: Importing by Management Class into the Target VSM System

It's a month later, and we're finally ready for the "receive" (import) portion of the export/import operation.

To import into a target VSM system, do the following:

1. **If the VTVs and MVCs you are importing are not in the target system CDS, redo your POOLPARM/VOLPARM definitions to add these volsers...**

   ...as described in *Configuring HSC and VTCS*. If you're actually merging or consolidating data centers, adding volsers is usually a gimme. If necessary, also increase the CDS size on the target VSM system. For more information, see *Configuring HSC and VTCS* or *Managing HSC and VTCS*.

   What if there were duplicate VTV volsers across the source and target systems? In general, do the following:

   - If the source system has more current VTVs with the same volsers as those on the target system, specify REPLACE(ALL).
   - If you are moving the VTVs from the source to the target system (first time export/import), specify REPLACE(NONE). In this case, you have to decide what to do with the duplicate VTVs on a case by case basis.

2. **Enter the MVCs for import into a target VSM system LSM.**

   For more information, see *Managing HSC and VTCS*. Can you see what's going on here? You actually want to get the MVCs physically in place before you use IMPORT to tell the CDS that it has some new MVCs and VTVs.

3. **Optionally, do a "validate" run of IMPORT:**

```
//IMPORT  EXEC PGM=SLUADMIN,PARM='MIXED' REGION=6M
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//REMOTE1     DD DSN=hlq.REMOTE1,DISP=SHR
//SLSPRINT    DD SYSOUT=*
//SLSIN       DD *
 IMPORT MANIFEST(MOVE1) NOUPDATE
```

FIGURE 2-2 IMPORT utility example: validation import run, HSC is active

FIGURE 2-2 shows example JCL to run the IMPORT utility where:

- The Manifest File is the export Manifest specified in Step 2 on on page 22.
- REPLACE(NONE) (the default) specifies that VTCS does not overwrite duplicate VTVs.
- IMMDRAIN(NO) (the default) specifies that VTCS does not drain all imported VTVs to VTSS space.
- NOUPDATE specifies that the CDS is not updated (validate run only).
- INACTCDS is not specified, so HSC is active.

Doing a validate run optional but **highly recommended**, because you really want to see what's going to happen before you push the button for real. Study carefully the Import Report. Like what you see? Continue with Step 4 on on page 25.

Note –

- IMPORT is valid only if FEATures VSM(ADVMGMT) is specified.
- Ensure that the "to" CDS has the same features (enabled by CDS level) as the "from" CDS. For example, if the “from” CDS has Large VTV page sizes enabled and 2/4 Gb VTVs have been created, then the "to CDS" must have the same capabilities, otherwise the import fails.

4. **Do an actual run of IMPORT:**

```
//IMPORT  EXEC PGM=SLUADMIN,PARM='MIXED' REGION=6M
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//REMOTE1     DD DSN=hlq.REMOTE1,DISP=SHR
//SLSPRINT    DD SYSOUT=*
//SLSIN       DD *
 IMPORT MANIFEST(MOVE1) REPLACE(ALL)
```

FIGURE 2-3 IMPORT utility example: validation import run, HSC is active

FIGURE 2-3 shows example JCL to run the IMPORT utility where, as in the “validate” run, REPLACE(NONE) (the default) specifies that VTCS does not overwrite duplicate VTVs.

Note – What if you want to return the MVCs to the source system? If so, you can specify IMMDRAIN(YES) to drain the import MVCs.

5. **Adjust your VTV definitions as needed.**

   For example, you need to define any new VTVs to the target system’s TMS.

6. **Do one of the following:**
   - Optionally, run MVCMAINT to make imported MVCs writable. VTCS imports MVCs as readonly. To make them writable, you run MVCMAINT, specifying READONLY OFF. The chances are that you are going to want to use the new MVCs at the target system, and this is the first step.

     Next, add the imported MVCs to the MVC pool as described in *Managing HSC and VTCS*. At this point, the MVCs can be reclaimed, drained, migrated to, recalled from, and so forth.
   - If you specified IMMDRAIN(YES) in Step 4, you can return the MVCs to the source system.

Exporting and Importing by Storage Class

Example: Exporting by Storage Class from the Source VSM System

This is the "send" phase of export/import, where we get the desired data packaged up and moved out of the source VSM system.

To export from a source VSM system, do the following:

1. **Identify the Storage Classes used for export.**
2. **Export by Storage Class:**

```
//EXPORT  EXEC PGM=SLUADMIN,PARM='MIXED' REGION=6M
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//MOVE2       DD DSN=hlq.REMOTE2,DISP=(,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
//SLSPRINT  DD SYSOUT=*
//SLSIN       DD *
 EXPORT STOR(OFF1,OFF2) MANIFEST(MOVE2)
```

FIGURE 2-4 EXPORT by Storage Class

In FIGURE 2-1, the output manifest file is MOVE2, which are required for the import. Because we exported by Storage Class, a Manifest File is created by no VTV consolidation occurs. The export MVCs are marked as read-only and as exported in the CDS, and are now available for ejection from a source system LSM.

The VTVs were resident on the MVCs removed from the LSM can still be used provided they are resident on other MVCs.

Caution – Schedule the export for a time when the exported data is not being updated!

---

3. **Remove the MVCs for export from the MVC pool.**

   For more information, see *Managing HSC and VTCS*.

4. **Eject the MVCs for export from a source VSM system LSM.**

   For more information, see *Managing HSC and VTCS.*

5. **If desired, at the source system, scratch or make unavailable the exported VTVs or reuse the data they contain.**

   After the export, the source system retains the CDS records of the exported VTVs and MVCs. The export MVCs are marked as exported and readonly in the source system CDS. At this point, you have two choices, depending on why you exported the VTVs:

   - **If you exported the VTVs to provide a backup copy at a second site**, leave the VTVs as readonly in the source system CDS so they cannot be updated.
   - **If you are permanently moving the exported VTVs to a second site**, then scratch or otherwise make them unavailable in the source system CDS. Use the HSC scratch utilities or the LCM SYNCVTV function to scratch exported VTVs.

Example: Importing by Storage Class into the Target VSM System

It's a month later, and we're finally ready for the "receive" (import) portion of the export/import operation.

To import into a target VSM system, do the following:

1. **If the VTVs and MVCs you are importing are not in the target system CDS, redo your POOLPARM/VOLPARM definitions to add these volsers...**

   ...as described in *Configuring HSC and VTCS*. If you're actually merging or consolidating data centers, adding volsers is usually a gimme. If necessary, also increase the CDS size on the target VSM system. For more information, see *Configuring HSC and VTCS* or *Managing HSC and VTCS*.

   What if there were duplicate VTV volsers across the source and target systems? In general, do the following:

   - If the source system has more current VTVs with the same volsers as those on the target system, specify REPLACE(ALL).
   - If you are moving the VTVs from the source to the target system (first time export/import), specify REPLACE(NONE). In this case, you have to decide what to do with the duplicate VTVs on a case by case basis.

2. **Enter the MVCs for import into a target VSM system LSM.**

   For more information, see *Managing HSC and VTCS*. Can you see what's going on here? You actually want to get the MVCs physically in place before you use IMPORT to tell the CDS that it has some new MVCs and VTVs.

3. **Optionally, do a "validate" run of IMPORT:**

```
//IMPORT  EXEC PGM=SLUADMIN,PARM='MIXED' REGION=6M
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//REMOTE1     DD DSN=hlq.REMOTE1,DISP=SHR
//SLSPRINT    DD SYSOUT=*
//SLSIN       DD *
 IMPORT MANIFEST(REMOTE1) NOUPDATE
```

FIGURE 2-5 IMPORT utility example: validation import run, HSC is active

FIGURE 2-2 shows example JCL to run the IMPORT utility where:

- The Manifest File is the export Manifest specified in Step 2 on on page 22.
- REPLACE(NONE) (the default) specifies that VTCS does not overwrite duplicate VTVs.
- IMMDRAIN(NO) (the default) specifies that VTCS does not drain all imported VTVs to VTSS space.
- NOUPDATE specifies that the CDS is not updated (validate run only).
- INACTCDS is not specified, so HSC is active.

Doing a validate run optional but **highly recommended**, because you really want to see what's going to happen before you push the button for real. Study carefully the Import Report. Like what you see? Continue with Step 4 on on page 25.

Note –

- IMPORT is valid only if FEATures VSM(ADVMGMT) is specified.
- Ensure that the "to" CDS has the same features (enabled by CDS level) as the "from" CDS. For example, if the “from” CDS has Large VTV page sizes enabled and 2/4 Gb VTVs have been created, then the "to CDS" must have the same capabilities, otherwise the import fails.

4. **Do an actual run of IMPORT:**

```
//IMPORT  EXEC PGM=SLUADMIN,PARM='MIXED' REGION=6M
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//REMOTE1     DD DSN=hlq.REMOTE1,DISP=SHR
//SLSPRINT    DD SYSOUT=*
//SLSIN       DD *
 IMPORT MANIFEST(REMOTE1)
```

FIGURE 2-6 IMPORT utility example: validation import run, HSC is active

FIGURE 2-3 shows example JCL to run the IMPORT utility where, as in the “validate” run, REPLACE(NONE) (the default) specifies that VTCS does not overwrite duplicate VTVs.

Note – What if you want to return the MVCs to the source system? If so, you can specify IMMDRAIN(YES) to drain the import MVCs.

5. **Adjust your VTV definitions as needed.**
6. **Do one of the following:**
   - Optionally, run MVCMAINT to make imported MVCs writable. VTCS imports MVCs as readonly. To make them writable, you run MVCMAINT, specifying READONLY OFF. The chances are that you are going to want to use the new MVCs at the target system, and this is the first step.

     Next, add the imported MVCs to the MVC pool as described in *Managing HSC and VTCS*. At this point, the MVCs can be reclaimed, drained, migrated to, recalled from, and so forth.
   - If you specified IMMDRAIN(YES) in Step 4, you can return the MVCs to the source system.

CHAPTER 3

Using the ELS External Vaulting Feature

The ELS External Vaulting Feature (ELS Vault) replaces and vastly improves upon its predecessor, the VSM Offsite Vault Feature. ELS Vault provides the following enhancements for vaulting real tape volumes:

- Uses the HSC CDS to store vault and vaulted volume data. Using CDS vaulting information instead of the TMS eliminates:
  - The risk of human error when returning volumes to the automated environment
  - Stranding volumes in the vault location when missed on the return pull list
  - Leaving accidentally returned vault volumes in the automated environment
- Uses LCM to manage the vaulting process, which has three vault methods:
  - Vaulting MVCs for Disaster Recovery (DR); for more information, see "Vaulting MVCs for DR" on page 33.
  - Vaulting MVCs for Long Term Retention (LTR); for more information, see "Vaulting MVCs for LTR" on page 45
  - Ejecting volumes to a floor vault; for more information, see "Ejecting Specific Volumes to a Local (Floor) Vault" on page 46.

Preparing for ELS External Vaulting

The first step is to define the vaulted volume area of the HSC CDS. To do so, you run the SLUADMIN SET VAULTVOL utility. For example:

```
SET VAULTVOL NBRVOLS(40000)
```

Note –

- If you need to add more vault volumes after the initial definition, you need to do so via MERGEcds, so allow sufficient number of volumes at initial definition to cover your anticipated needs.
- You must have sufficient FREE blocks in your CDS to accommodate the volumes you plan to vault (real tape volumes) plus some overhead for growth. See *Configuring HSC and VTCS* for information on calculating the space for vault volumes and see *Managing HSC and VTCS* for how to expand the CDS if the CDS is not big enough to hold your vaulted volumes.

Step two is to define the vaults that contain your vaulted volumes. To do so, you run the SLUADMIN SET VAULT utility for each vault. For example:

```
SET VAULT ADD NAME(DRVLT1) SLOTS(10000) DESC('DR Vault')
SET VAULT ADD NAME(LTRVLT1) SLOTS(20000) DESC('LTR Vault')
SET VAULT ADD NAME(FLOOR) SLOTS(500) DESC('Floor Vault')
```

Note – The total number of slots in all the vaults that you define cannot exceed the number of volumes specified in the VAULTVOL statement.

While HSC defines the vaults and vaulted volumes, LCM manages them. Note especially the following LCM vaulting parameters:

GRACEPERIOD

The number of days between when a volume is selected for return from the vault and when it is actually returned to the automated environment. The grace period provides a safety margin so that new volumes arrive in the vault before the old volumes are returned. The default value if not specified is three days.

DEFAULT

DEFAULT is mutually exclusive with GRACEPERIOD and is typically used for the "floor" (manual rack) vault that automatically contains all volumes ejected from the automated environment via an LCM EJECT(ASNEEDED). Other volumes being ejected from the ACS can be assigned to this vault as well, for example, a volume that is active but no longer a good automation candidate. Generally this would be a defined vault that is actually racks on the datacenter floor. DEFAULT is a grace period of zero days to allow these volumes to be re-entered into the ACS at anytime.

Also note that standard eject options for are available for all vault volumes. This includes which CAPs to use, defining an eject message, mode for the ejects and whether the ejects are to be in slot or volume serial order.

Creating MVCs for DR and LTR

Whenever MVCs are vaulted for DR, you should make a **minimum** of two copies of each VTV to separate MVCs, one of which remains onsite while the other MVC is ejected and put in an external vault. This is done by assigning two Storage Classes on the Management Class that is assigned to the VTV.

So the challenge is, you want to safeguard your data but you want to use MVC space as economically as possible, as follows:

- **Define as few Storage Classes as possible.** Too many Storage Classes usually means too many MVCs and/or MVCs with few VTVs.
- **Use as few VTSSs as possible to create MVCs.** If at all possible, use one VTSS only to create vault MVCs.

What are other considerations for creating and vaulting MVCs? Consider the following:

- **First, VTVs need to migrate to vault MVCs as soon as possible** so that they can be ejected and moved to the vault because these VTVs are generally not used as input for other job steps.
- **Second, because DR VTVs expire at different rates**, consider grouping VTVs with similar expiration dates on MVCs. However, to reduce the total number of MVCs being created to send to the vault, limit the number of these groups. As there will already be activity to consolidate VTVs to fewer MVCs, the benefit is minimal beyond two groups, one for VTVs with very short expiration periods, like within the next 7 days, and one for all other volumes. VTVs under catalog control should be considered to be part of the second group as there is no way to know the actual expiration date.
- **Third, while it is not necessary to have separate vaults for DR and LTR** purposes, it may be advantageous to have the DR MVCs located in a separate vault. This allows these volumes to be gathered and sent to the DR site when needed.

  For LTR data the considerations are slightly different. First, LTR data by definition will not expire for an extended period. Thus, unlike DR data which expires over a period of time, LTR MVCs will not become fragmented. There will be initial periodic processing of these MVCs to get the vaulted MVCs to be as full as possible, just as with the DR MVCs, but once sufficiently full, these MVCs will remain static. There is no need to have more than a single storage class for these volumes. However, note that you may want to migrate immediately some LTR data, other data can be migrated as VTCS automigration selects it. Therefore, you may want to have one Storage Class but two Management Classes for LTR data.

DELSCR Considerations When Using the Vaulting Feature

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 and LTR Management Classes. If you specify DELSCR(YES), **only** use the LCM SYNCVTV for scratch synchronization. For more information about using LCM to manage scratch synchronization, see *LCM User's Guide*.

What Happens When Vaulted Volumes Return to the ACS?

HSC enter process has been modified for the ELS vaulting function so that every volume being entered is checked to determine if the volume is an externally vaulted volume. For such vaulted volumes, one of two actions will take place based upon the return date field in the CDS vault record:

- If the return date has occurred, the volume is entered, the stored volume metadata from the eject process is restored, and the volume is removed from the vault records.
- If the return date has **not** occurred or no return date has been set for the volume, the volume is entered, the stored volume metadata from the eject process is restored, but the volume is left in the vault records and it will automatically be ejected by the next eject process. Why would this occur? There are several reasons, where the two most common are that the volume was returned for some sort of data recovery process or the incorrect volume was pulled from the vault (a very common occurrence). Whatever the reason, the volume belongs in the vault and will be returned to that location and resume its protected status.

Note that this is the process for volumes vaulted in a *physical* vault. For volumes vaulted in a remote library, it's slightly different; see "DR Vaulting with MVCs in a Remote Library" on page 43.

Vaulting MVCs for DR

In a DR scenario, we have the overall business goals of optimizing VTSS buffer use, ensuring speedy migration of critical data, while maintaining data availability.

Basic DR Vaulting with MVCs

In this approach, DR volumes are created every day, thus the processing to move them to the DR vault runs every day to ensure that the MVCs are safely moved offsite and protected.

Note –

- All VTVs created for DR purposes and all native tapes, including the Manifest File tape created in "Step 2 – Exporting the Vault MVCs" on page 34, are controlled by the site's TMS (for volumeexpiration).This process is limited to the vaulting of the related MVCs and those selected native volumes involved.
- MVCs can only be assigned to one vault. To assign a volume to a new vault, the volume must first be removed from a prior vault assignment.
- "Step 7 – Prepare Vaulted MVCs for Return" on page 38 begins the periodic processing, which allows recycling of DR MVCs that have become fragmented or were only partially filled at creation. This is done by performing "logical" MVC drains of the vaulted MVCs using the copies of the still current VTVs located on the local MVCs. The periodic processing minimizes the number of total volumes in the vault while ensuring that overall related activity is reduced to a minimum, which is done by using selection criteria appropriate to the specific environment. After a successful "logical" drain of a vaulted MVC, the return date is set in the CDS for that volume. In the case of native volumes, such as the Manifest File tapes, volumes that have gone to a scratch status in the TMS are selected and set to be returned. You decide how often to run the periodic processing, ranging from daily to monthly.

Step 1 – Creation of Vault VTVs/MVCs

DR Vault VTVs are created using a Management Class that points to two Storage Classes, one that creates MVCs that will remain within the local environment and one that creates MVCs that are vaulted. For example:

```
STOR NAME(DRLOC)ACS(00) MEDIA(STK1RD)
STOR NAME(DRVLT1) ACS(00) MEDIA(STK1RD)
```

Step 2 – Exporting the Vault MVCs

You export Vault MVCs via an LCM parameter file such as the example shown in CODE EXAMPLE 3-1.

```
Options
  NoSync
  NoTMS
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Export
  Control(Serial )
  MVC
  DSN(DRVAULT.MANIFEST)
  Storageclass(DRVLT1)
  Vault('DRVLT')
  ;
```

CODE EXAMPLE 3-1 Exporting Vault MVCs

In CODE EXAMPLE 3-1:

- The OPTIONS statement specifies NOSYNC and NOTMS because vaulting does not use TMS information for vaulting and no TMS metadata is required.
- The VAULT statement specifies DRVLT as the DR vault.
- The ACTION EXPORT statement specifies:
  - Export MVCs by volser.
  - Create an Export Manifest File (DRVAULT.MANIFEST), in this case, a volume in the ACS that is ejected and stored with the DR MVCs.
  - Point to the vault Storage Class created in "Step 1 – Creation of Vault VTVs/MVCs" on page 33.
  - Define the DR Vault (DRVLT) and assign MVCs to it not previously assigned to the vault.

Note – Exported MVCs are marked read-only by the export processing.

You can create multiple Vault Storage Classes (for example, to segregate MVCs with VTVs with different expiration dates). If you want to assign different Vault Storage Classes to the same vault, you can do so in a single ACTION EXPORT statement. For example, the following statements assigns Storage Classes DRVLT1 and DRVLT2 to the same vault (DRVLT):

```
Action
  Export
  Control(Serial )
  MVC
  DSN(DRVAULT.MANIFEST)
  Storageclass(DRVLT1
               DRVLT2)
  Vault('DRVLT')
  ;
```

Step 3 – (Optional) Write Additional Datasets to Manifest File Tape

After you create the Manifest File tape in "Step 2 – Exporting the Vault MVCs" on page 34, you can run a job that copies the HSC Control Data Set (CDS), the TMS catalog, the system catalog, and other significant "point in time" data sets to the Manifest File tape to provide additional DR recovery points.

Step 4 – Eject Vault MVCs

You eject Vault MVCs via an LCM parameter file such as the example shown in CODE EXAMPLE 3-2.

```
Options
  NoSync
  NoTMS
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Eject
  When(
  (InLsm)
   and
  (VaultName EQ 'DRVLT')
  Control(Serial)
  Ejmsg('Move to DR Vault')
  ;
```

CODE EXAMPLE 3-2 Ejecting Vault MVCs

In CODE EXAMPLE 3-2:

- The OPTIONS statement specifies NOSYNC and NOTMS because vaulting does not use TMS information for vaulting and no TMS metadata is required.
- The VAULT statement specifies DRVLT as the DR vault.
- The ACTION EJECT statement specifies:
  - Eject MVCs assigned to DRVLT; eject by volser.
  - The eject message.

Step 5 - Eject Native Volumes (Including Manifest File Tape)

You eject native volumes (including the Manifest File tape) via an LCM parameter file such as the example shown in CODE EXAMPLE 3-3.

```
Options
  NoSync
  ;
TMS
  RMM
  Dateform(J)
  DDname(LCMTMSDB)
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Eject
  When(
  (InLsm)
   and
  (DataSetName EQ 'DRVLT.MANIFEST')
   and
  (TMSScratch EQ False)
     )
  Control(Serial)
  Ejmsg('Move to DR Vault')
   ;
```

CODE EXAMPLE 3-3 Ejecting Native Volumes (Including Manifest File Tape)

In CODE EXAMPLE 3-3:

- The OPTIONS statement specifies NOSYNC because vaulting does not use TMS information for vaulting.
- The VAULT statement specifies DRVLT as the DR vault.
- The ACTION EJECT statement specifies:
  - Eject the Manifest File tape.
  - Eject any native volumes not scratched by the TMS
  - The eject message.

Step 6 – Create a Pull List of Volumes for Return from the Vault

You create a Pull List via an LCM parameter file such as the example shown in CODE EXAMPLE 3-4.

```
Options
  NoSync
  NoTMS
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Report
  Volume
  Sysout(*)
  Title('Return Report')
  When(
  (VaultName EQ 'DRVLT')
   and
  (VaultReturnDate LE TODAY)
   and
  (VaultReturnDate NE MISSING)
     )
  Column (Serial,
          VaultSlot)
  ;
```

CODE EXAMPLE 3-4 Creating a Pull List

In CODE EXAMPLE 3-4:

- The OPTIONS statement specifies NOSYNC and NOTMS because vaulting does not use TMS information for vaulting and no TMS metadata is required.
- The VAULT statement specifies DRVLT as the DR vault.
- The REPORT VOLUME statement creates a report that lists the volumes in the vault that have reached their previously assigned Return Dates. This is a simple example, you can add more selection criteria for the volumes you want returned.

Note –

- 'TODAY' and 'MISSING' are unique values when considering dates. 'TODAY' is translated to the date of the LCM run execution. 'MISSING' means that there is no value for the date. In this example, it means that no date has been set. Both conditions are required as a missing date would be seen as 'Less Than' the current date.
- Steps 4, 5 and 6 can be combined into a single jobstep. In some cases, Step 6 is executed on a periodic basis, generally the day before any volumes would be returned from the vault.

Step 7 – Prepare Vaulted MVCs for Return

You prepare Vaulted MVCs for return via an LCM parameter file such as the example shown in CODE EXAMPLE 3-5.

```
Options
  NoSync
  NoTMS
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Drain
  When(
  (MVC EQ True)
   and
  (VaultName EQ 'DRVLT')
   and
  (MVCVTVCount LE 30)
   and
  (MVCInUse LE 30)
   and
  (Days_Since(VaultAssignmentDate) GT 7)
  )
  Control(MVCVTVCount
        Ascending)
  Limit(30)
  ;
```

CODE EXAMPLE 3-5 Preparing Vaulted MVCs for Return

In CODE EXAMPLE 3-5:

- The OPTIONS statement specifies NOSYNC and NOTMS because vaulting does not use TMS information for vaulting and no TMS metadata is required.
- The VAULT statement specifies DRVLT as the DR vault.
- For MVCs currently in the DR vault, the ACTION DRAIN statement specifies draining MVCs that:
  - Have fewer than 30 VTVs.
  - Are in use less than 30% of the time.
  - Have been in the vault at least 7 days.
  - Limit the number of returned MVCs to a maximum of 30.
  - The return date is set to 3 days by the GracePeriod parameter.

  When you create a parameter file to drain MVCs, you want to balance between the processing cycles for the drain and the need to recycle fragmented or partially filled MVCs. Accordingly, the MVCVTVCount, MVCInUse, Days_Since, and LIMIT parameters provide the controls to balance these requirements.

Step 8 – Prepare Vaulted Native Volumes for Return

You prepare native volumes for return via an LCM parameter file such as the example shown in CODE EXAMPLE 3-6.

```
Options
  NoSync
  ;
TMS
  RMM
  Dateform(J)
  DDname(LCMTMSDB)
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Vault
  Return
  When(
  Not (MVC)
   and
  (VaultName EQ 'DRVLT')
   and
  (TMSScratch EQ True)
      )
  ;
```

CODE EXAMPLE 3-6 Preparing Vaulted Native Volumes for Return

In CODE EXAMPLE 3-6:

- The OPTIONS statement specifies NOSYNC because vaulting does not use TMS information for vaulting.
- The TMS RMM statement is necessary to add the processing of TMS metadata.
- The VAULT statement specifies DRVLT as the DR vault.
- The ACTION VAULT RETURN statement sets a return date (via the GracePeriod parameter) for volumes that are not MVCs and are in scratch status in the TMS.

Step 9 - Entering Returned Volumes

Volumes that appear on the report created in "Step 6 – Create a Pull List of Volumes for Return from the Vault" on page 37 are removed from the DR vault and returned to the local environment. When you enter these volumes in the ACS, HSC checks if the assigned Vault Return Date has occurred for each volume. If it has, the volume has attained its scheduled return date, and it is removed from the vault. Returned MVCs are then made eligible for migration and native volumes are made scratch in the CDS the next time that LCM SYNC processing occurs. If the Vault Return Date has **not** occurred, the volume is ejected by "Step 4 – Eject Vault MVCs" on page 35.

If desired, you can combine Steps 7 and 8 in a single LCM parameter file as shown in the example in CODE EXAMPLE 3-7.

```
Options
  NoSync
  ;
TMS
  RMM
  Dateform(J)
  DDname(LCMTMSDB)
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Drain
  When(
  (MVC EQ True)
   and
  (VaultName EQ 'DRVLT')
   and
  (MVCVTVCount LE 30)
   and
  (MVCInUse LE 30)
   and
  (Days_Since(VaultAssignmentDate) GT 7)
  )
  Control(MVCVTVCount
        Ascending)
  Limit(30)
  ;
Action
  Vault
  Return
  When(
  Not (MVC)
   and
  (VaultName EQ 'DRVLT')
   and
  (TMSScratch EQ True)
     )
  ;
```

CODE EXAMPLE 3-7 Preparing All Volumes for Return

Multiple Week DR Vaulting with MVCs

Some sites may elect to use a multiple week process where DR processing includes a weekly full volume backup of critical data, followed by daily incremental backups over the next six days. The external vaulting process moves volumes offsite on day of creation. This process assumes a four week cycle that keeps the DR data offsite and complete until the fourth week starts. **Note that** the only change between "Basic DR Vaulting with MVCs" on page 33 and the multiple week process is the difference in selection criteria for Steps 7 and 8 as described below. This is done by setting the expiration dates so that the vault VTVs on the associated MVCs and the Manifest File tapes (and any other native tapes involved) all expire on the specific date 22 days after creation.

The multiple week timeline is as follows:

- Day 1 – Full Volume Backups (Expire on Day 22).
- Day 2 – Incremental Backup #1 (Expire on Day 22).
- Day 3 – Incremental Backup #2 (Expire on Day 22).
- Day 4 – Incremental Backup #3 (Expire on Day 22).
- Day 5 – Incremental Backup #4 (Expire on Day 22).
- Day 6 – Incremental Backup #5 (Expire on Day 22).
- Day 7 – Incremental Backup #6 (Expire on Day 22).
- Days 8 through 21 – volumes remain off site.
- Day 22 – Backups and Manifest File tapes from Days 1 through 7 expire and VTVs go scratch in CDS via the LCM VTVSYNC process. MVCs are drained using the following parameter in the selection criteria for "Step 7 – Prepare Vaulted MVCs for Return" on page 38 and "Step 8 – Prepare Vaulted Native Volumes for Return" on page 39

DAYS SINCE (VaultAssignmentDate) GT 15

The return date for drained MVCs and Manifest File tapes is set for Day 25.

Note – All MVCs assigned to the vault during the first seven days of the cycle are drained. If the expiration date of the DR VTVs was set correctly, there should be no current VTVs at this point and the logical drain process will execute quickly. If current VTVs are written to a new MVC, the cause would be an incorrect expiration date being set.

- Day 23 and 24 – Volumes remain offsite.
- Day 25 – Vaulted volumes from Days 1 through 7 are returned and removed from vault status and are available for reuse.
- Day 29 – Cycle repeats.

Note – Some sites may elect to only take the full volume backups off site, keeping the incremental backups on site. In this case, place the incremental MVCs and the related Manifest File and native volumes in a container that is locked and not reopened until its return on Day 25. At that point, all volumes created on Day 1 should be expired.

DR Vaulting with MVCs in a Remote Library

In this process, instead of vaulting volumes in a physical vault, the volumes are vaulted in a remote library (ACS). This process is similar to "Basic DR Vaulting with MVCs" on page 33 with the following exceptions:

- Steps 1 through 3 - No change.
- Step 4 - Eliminated because no ejects of vaulted MVCs are done.
- Step 5 - As with Step 4, native volumes are not ejected. Instead, an Action Vault Assign statement assigns native volumes to the vault using the same selection criteria that would have been used to perform and eject of these native volumes as shown in the example in CODE EXAMPLE 3-8.

```
Options
  NoSync
  ;
TMS
  RMM
  Dateform(J)
  DDname(LCMTMSDB)
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Vault
  Assign
  Vault('DRVLT')
  When(
  (InLsm)
   and
  (DataSetName EQ 'DRVLT.MANIFEST')
   and
  (TMSScratch EQ False)
     )
  ;
```

CODE EXAMPLE 3-8 Assigning Native Volumes to the Vault

- Step 6 - Eliminated because no volumes are reentered.
- Step 7 - No change.

Note – Because the drain processing occurs within the remote library, drains execute more efficiently than the logical drain processing for volumes in a manual vault.

- Step 8 - No change.

- Step 9 - In the Basic Process, the vault assignment is removed by the enter processing which does not occur in the remote library scenario. To remove the vaulted volume, use the Action Vault Release statement. The vaulted volumes to be released are selected by Vault Name and the Return Date as was previously done to generate a Pull List. The Action Vault Release statement only processes vaulted volumes whose Return Date has occurred.

  CODE EXAMPLE 3-9 shows an example of the Action Vault Release statement.

```
Options
  NoSync
  NoTMS
  ;
Vault
  Name('DRVLT')
  GracePeriod(3)
  ;
Action
  Vault
  Release
  When(
  (VaultName EQ 'DRVAULT')
   and
  (VaultReturnDate LE TODAY)
   and
  (VaultReturnDate NE MISSING)
     )
  ;
```

CODE EXAMPLE 3-9 Removing Vault Assignment with Action Vault Release

Vaulting MVCs for LTR

The process for using MVCs to hold Long Term Retention (LTR) MVCs is essentially the same as described in the Basic Process for Disaster Recovery described in "Basic DR Vaulting with MVCs" on page 33. The two major considerations for LTR usage are:

- Movement to the vault, "Step 2 – Exporting the Vault MVCs" on page 34 through "Step 5 - Eject Native Volumes (Including Manifest File Tape)" on page 36, may not be a daily function so that more LTR volumes are completely filled before being vaulted.
- The LTR VTVs will not expire for long periods of time, thus the periodic process described may only occur at longer time intervals. There will still be LTR MVCs that are only partially filled, so at some interval, those partially filled MVCs with fewer VTVs and low MVC usage must be processed so that the VTVs on the partially filled MVCs are consolidated on fewer LTR MVCs.

At some point in the future, there may be a need perform the logical drains of the LTR MVCs to move the archived data to new media. The basic process easily performs that activity by simply making the appropriate selection criteria choices and limiting the number of vaulted LTR MVCs processed at one time. With each execution, the still current VTVS are moved to the new media, those new MVCs are moved to the vault and the logically drained MVCs are returned for reuse or destruction.

Ejecting Specific Volumes to a Local (Floor) Vault

Many sites may require specific volumes to be removed from the automated environment and need to efficiently store these volumes locally in racks in the local environment. This requirement may come from such activities as retiring volumes from use and desiring to keep the volumes available for some period of time. Multiple local/floor vaults can be defined with specific volumes being sent to different vaults, but only one local/floor vault can be defined as a 'Default' vault. All volumes placed in a local/floor vault are automatically assigned a Return Date of the current date. This allows these volumes to be returned to the automated environment and deleted from their vault assignment with no further action.

CODE EXAMPLE 3-10 shows an LCM parameter file example of ejecting specific volumes to a Floor Vault.

```
Options
  NoSync
  ;
TMS
  RMM
  Dateform(J)
  DDname(LCMTMSDB)
  ;
Vault
  Name('FLOOR')
  Default
  ;
Action
  Eject
  When(
  (InLsm EQ True)
   and
  (DaysSinceReference GT 100)
   and
  (MVC EQ False)
   and
  Not
  (DataSetName Matches 'HMIG.**')
     )
   Control(
        VaultSlot
        Ascending
        )
  Ejmsg('Move to Floor Vault')
;
Manage
  ACSID(00)
  Numfree(500)
  ;
```

CODE EXAMPLE 3-10 Ejecting Specific Volumes to a Floor Vault

In CODE EXAMPLE 3-10 on page 46:

- The OPTIONS statement specifies NOSYNCH because vaulting does not use TMS information for vaulting.
- The TMS RMM statement is necessary to add the processing of TMS metadata.
- The VAULT statement specifies FLOOR as the default floor vault.
- The ACTION EJECT statement specifies to eject the volumes to the Floor Vault that:
  - Are in the LSM.
  - Have not been referenced in over 100 days.
  - Are **not** MVCs.
  - Have a data set name mask of HMIG.**.
- The ACTION EJECT statement also specifies:
  - Process volumes in ascending volser order.
  - Eject by TMS slot numbers.
  - The eject message.
- The MANAGE statement specifies:
  - Manage volumes in ACS 00.
  - Ensure 500 free cells in the ACS.

CHAPTER 4

Using Cross-TapePlex Replication in a DR Solution

Back in "Doing Physical Exports and Imports" on page 21, we explained how to create "export" portable MVCs from a source site so you can physically move these MVCs to a target site and import the MVCs (and the VTVs they contain) into the target site. Now we're going to talk about *Cross-TapePlex Replication (CTR)*. So what's the difference? Well, with CTR, you no longer use the PTAM (Pickup Truck Access Method) to move MVCs from one site to another. Instead, you move VTVs electronically from source to target site...that is, from one TapePlex to another...where the VTVs are then migrated to MVCs, which eliminates the PTAM step. As a copy of the VTV moves from the source to the target TapePlex, a copy of the VTV's metadata moves from the source TapePlex's CDS to the target TapePlex's CDS. **Note that** the source TapePlex continues to "own" and manage the scratching of the CTR VTVs.

Caution – Note that if you are using CTR, stopping SMC stops VTCS from sending metadata to a CTR TapePlex, which effectively stops data transfer. Therefore, if you are using an HSC feature that uses the SMC communication services, such as CTR, then you should ensure that HSC activity is quiesced or HSC is terminated before stopping SMC.

How Does CTR Work?

Let's take a look at a CTR as shown in FIGURE 4-1.

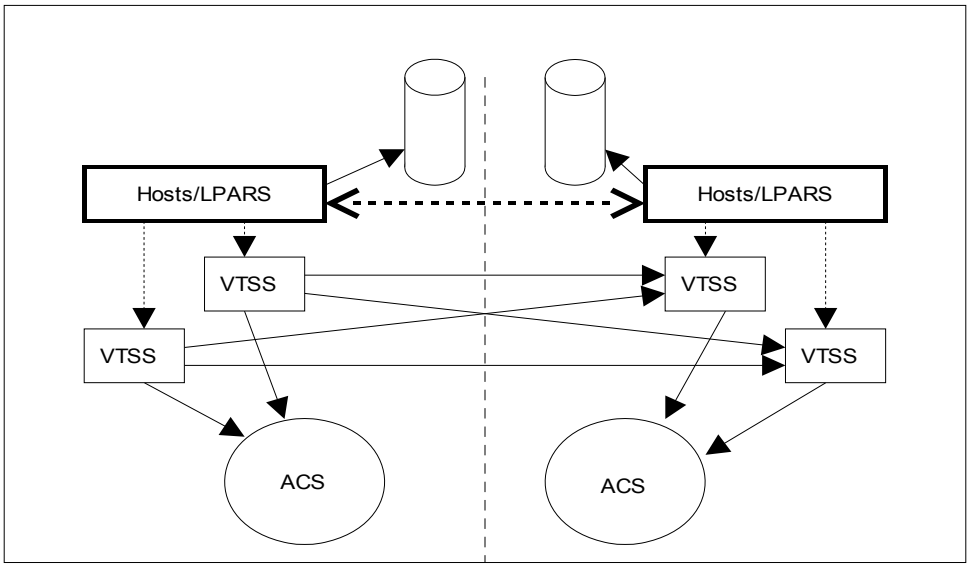

FIGURE 4-1 ELS CTR Configuration

As FIGURE 4-1 shows:

- CTR uses the connection between two VTSSs (CLINK) in two separate TapePlexes to send data from one VTSS to another. The connection can be either uni-directional or bi-directional.
- CTR uses the services of the SMC client/server feature to send metadata from the sending TapePlex to the receiving TapePlex. Note that you do NOT need to use the client/server feature to communicate between SMC and HSC in order to use CTR, but you must define HTTP and SERVER commands in SMC to allow metadata to be transferred.
- There are separate (and separately maintained) CDSs at each site, so that loss of connectivity or hardware availability within one site does not directly affect any of the other sites.
- The configuration and physical connection requirements are simple and straightforward.
- You can now run concurrent DR tests more simply and without disruption to existing work (without using the CDRT utility).
- You can now do an automatic switchover of workload from one site to another.
- The VTV volume ranges for the two TapePlexes are per FIGURE 4-2. **Note that** each TapePlex has its own set of writable volumes and that these are mirrored on the other TapePlex by read-only versions.
- The configuration shown has both VTSSs at the sending TapePlex connected to both VTSSs at the receiving TapePlex for maximum resilience.

Note – In both Clustered VTSS and CTR configurations, you must ensure that the first 16 VTDs in each VTSS (0-F) are reserved for replication. These devices must be OFFLINE to MVS, and their paths must be online to each HSC server host. VTCS does not register the first 16 VTDs with SMC/HSC, which prevents mounting VTVs on these VTDs.

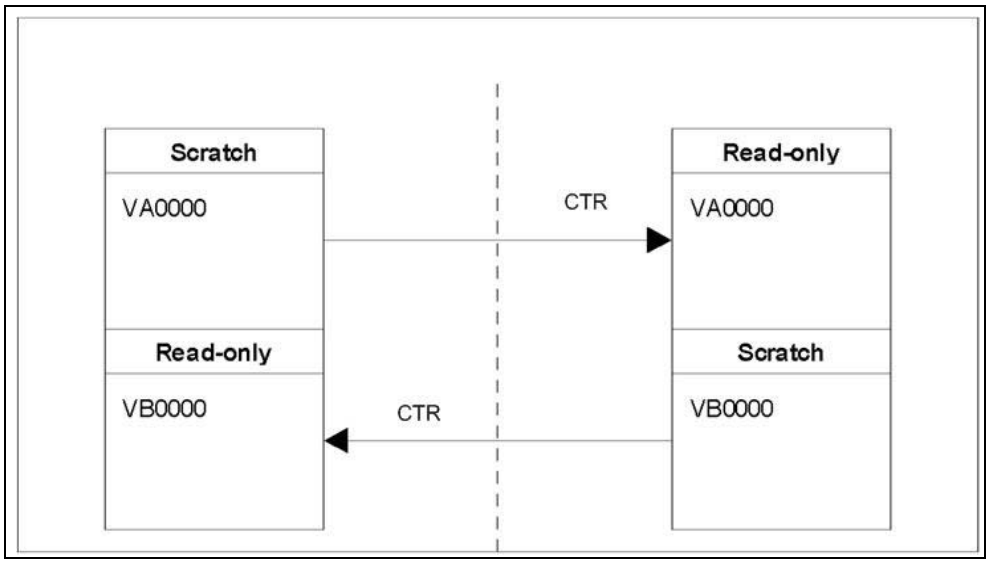

FIGURE 4-2 Intra-Site VTV Volume Relationships

Sounds great, doesn't it? First, read "CTR VTV Read-Only Considerations" on page 52, then go to "Configuring for CTR" on page 53...

CTR VTV Read-Only Considerations

When you use CTR, all VTVs replicated from one site to the other are in read-only mode at the remote site. Although these VTVs can be scratched (and their respective volume serial numbers re-used) by the remote TapePlex in the event of an actual disaster, their read-only status cannot be changed as long as the volumes are not in a SCRATCH status. Note that volumes in a POOLPARM EXTERNAL pool can never be set to SCRATCH status.

Therefore, if you elect to use CTR as a business continuance or disaster recovery strategy, you must ensure that your applications do not attempt to update these volumes, either during a DR test or in an actual disaster. The following scenarios should be considered:

1. Applications that use attribute DISP=MOD in JCL or dynamic allocation to append data to an existing data set must implement a checkpoint/restart mechanism and must record a checkpoint before creating any DISP=MOD volumes. These applications are recovered by restarting at the checkpoint and, if appropriate, must recreate these DISP=MOD volumes once restarted. Note that the use of DISP=MOD itself is not an issue with cross TapePlex replication. As long as the application has a checkpoint that allows it to back out partial updates, or a design that allows output of new data to begin on a new volume, it should run against read-only VTVs with no issues.
2. If VTVs that are replicated to another TapePlex are owned by HSM, the following process allows collection of data to start on a new volume and avoids updating existing HSM VTVs:
   a. Mark existing volumes full.
   b. Modify ARCCMD if necessary for USERUNITTABLE, MIGRATION, BACKUP, and RECYCLE.
   c. Ensure RECYCLEDALLOCFREQ is set to 1. This allows HSM allocation to allocate a new volume and device when appropriate.
   d. Depending on your MGMTCLAS VTVSIZE, set PERCENTFULL:
      i. For 800 MB VTVs set HSM PERCENTFULL to 97.
      ii. For 4 GB VTVs set HSM PERCENTFULL to 450

Applications that stack data sets on an existing volume are subject to the same DISP=MOD restrictions as above.

Configuring for CTR

FIGURE 4-3 shows an example of a CTR Configuration. In this system, VTSS VTSSA resides in TapePlex TAPEPLXA and has "partner" CLINKS to VTSS VTSSB in TapePlex TAPEPLXB. VTVs replicated to VTSSB are now resident in TAPEPLXB's CDS, as are the MVCs to which the VTVs are subsequently migrated. That is, VTVs are replicated across TapePlexes, then migrated locally. VTSSs in the sending TapePlex **cannot have connections** to RTDs in the receiving TapePlex.

Note – The following example shows a uni-directional CTR To do a bi-directional CTR, you simply define the configuration and SMC client/server control statements the same way on both TapePlexes. Note that a single TapePlex can also receive VTVs from multiple other TapePlexes. To define a configuration where one TapePlex is receiving data from multiple other TapePlexes, you simply add additional TapePlex names to the CONFIG of TAPEPLXB.

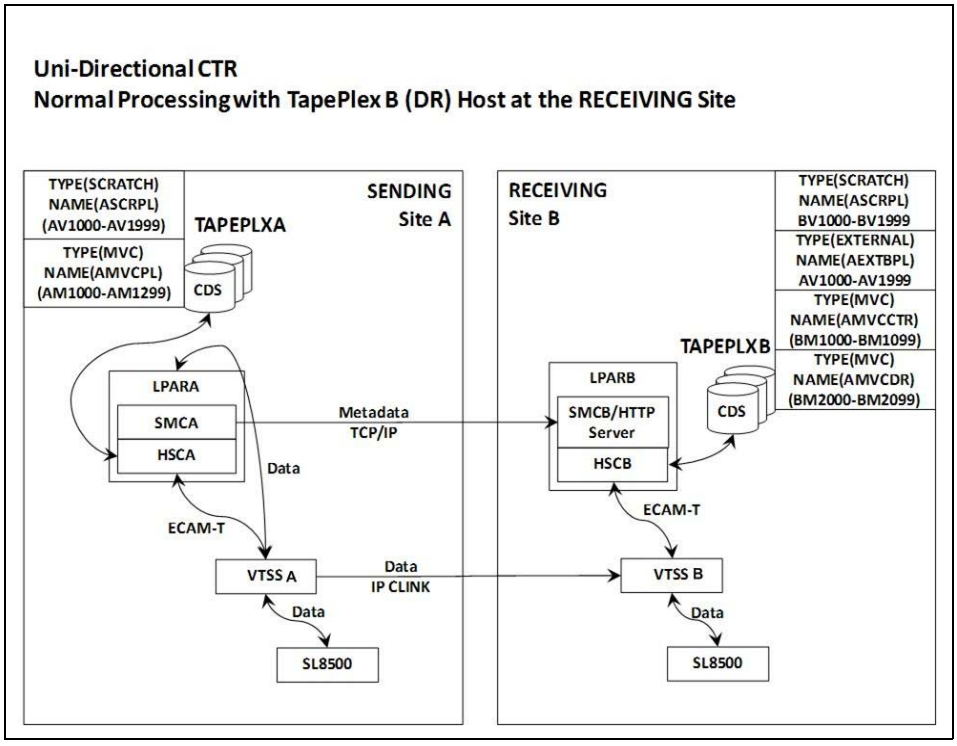

FIGURE 4-3 CTR Configuration

The Setup: Configuring and Starting CTR

To configure and start the example CTR system shown in FIGURE 4-3 on page 53, do the following:

1. **Ensure that your system has the Clustered VTSS requirements described in *Installing ELS*.**
2. **Start the HTTP server under the SMC running on host LPARB.**

   You may want to do this in your SMC CMDS file. For example:

   ```
   HTTP START PORT(999)
   ```

3. **Define your TAPEPLEX and SERVER commands on host LPARB.**

   Again, you may want to do this in your SMC CMDS file. For example:

   ```
   TAPEPLEX NAME(TAPEPLXA) LOCSUB(HSCA)
   TAPEPLEX NAME(TAPEPLXB)
   SERVER NAME(REMB)TAPEPLEX(TAPEPLXB) HOSTNAME(LPARB) PORT(999)
   ```

   Note – In the example configuration, TapePlex TAPEPLXB exists (from the perspective of TapePlex TAPEPLXA) for the sole purpose of maintaining a CDS containing metadata about VTVs that have been replicated from TAPEPLXA. However, if HSC or VTCS definitions in the TapePlexes TAPEPLXA and TAPEPLXB use the same device addresses referencing different physical devices, you must define SMC UNITATTR commands to tell SMC which TapePlex defines the devices on its host. Although the UNITATTR must specify a MODEL, if the specified model does not match the model reported by the TapePlex, the actual model overrides the UNITATTR MODEL. The following is an example of the SMC UNITATTR statement that would be used if the TapePlexes TAPEPLXA and TAPEPLXB both defined the address range 9000-90FF:

   ```
   UNITATTR ADDR(9000-90FF)  TAPEPLEX(TAPEPLXA) MODEL(VIRTUAL)
   ```

4. **Code a CONFIG deck for TapePlex A, as shown in FIGURE 4-4 on page 55.**

   In this figure, note:

   - The TAPEPLEX statement, which defines this TapePlex
   - The CLINK statements define the CLINKs that are used for CTR from VTSSA to VTSSB.
   - The Conditional Replication setting on the CONFIG GLOBAL statement is CHANGED for TAPEPLXA.

5. **Code a CONFIG deck for TapePlex B, as shown in FIGURE 4-5 on page 55.**

   In this figure, note:

   - The TAPEPLEX statement includes a RECVPLEX=TAPEPLXA parameter to specify that TAPEPLXB can receive VTVs from TAPEPLXA.
   - There are no CLINK statements, because the CLINKs are defined in the CONFIG deck for TAPEPLXA.

```
//CREATCFG      EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB    DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL   DD DSN=hlq.TAPEPLXA.DBASEPRM,DISP=SHR
//SLSCNTL2 DD DSN=hlq.TAPEPLXA.DBASESEC,DISP=SHR
//SLSSTBY   DD DSN=hlq.TAPEPLXA.DBASESBY,DISP=SHR
//SLSPRINT  DD   SYSOUT=*
//SLSIN       DD   *
 CONFIG RESET CDSLEVEL(V62ABOVE)
 GLOBAL MAXVTV=65000 MVCFREE=60 VTVATTR=SCRATCH RECALWER=YES
 LOCKSTR=STK_VTCS_LOCKS VTVPAGE=LARGE REPLICAT=CHANGED
 RECLAIM THRESHLD=70 MAXMVC=30 START=98 CONMVC=1
TAPEPLEX THISPLEX=TAPEPLXA
VTSS NAME=VTSSA LOW=71 HIGH=80 MAXMIG=8 MINMIG=1 RETAIN=10
 RTD  NAME=VSMA1A00 DEVNO=1A00 CHANIF=0C
 RTD  NAME=VSMA1A01 DEVNO=1A01 CHANIF=0D
 RTD  NAME=VSMA1A02 DEVNO=1A02 CHANIF=0K
 RTD  NAME=VSMA1A03 DEVNO=1A03 CHANIF=0L
 RTD  NAME=VSMA2A08 DEVNO=2A08 CHANIF=1C
 RTD  NAME=VSMA2A09 DEVNO=2A09 CHANIF=1D
 RTD  NAME=VSMA2A0A DEVNO=2A0A CHANIF=1K
 RTD  NAME=VSMA2A0B DEVNO=2A0B CHANIF=1L
 VTD LOW=7900 HIGH=79FF
VTD LOW=8900 HIGH=89FF
 CLINK VTSS=VTSSA CHANIF=0G REMPLEX=TAPEPLXB PARTNER=VTSSB
 CLINK VTSS=VTSSA CHANIF=0O REMPLEX=TAPEPLXB PARTNER=VTSSB
```

FIGURE 4-4 CONFIG for CTR Example - TapePlex TAPEPLXA

```
//CREATCFG       EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB     DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL    DD DSN=hlq.TAPEPLXB.DBASEPRM,DISP=SHR
//SLSCNTL2  DD DSN=hlq.TAPEPLXB.DBASESEC,DISP=SHR
//SLSSTBY    DD DSN=hlq.TAPEPLXB.DBASESBY,DISP=SHR
//SLSPRINT   DD   SYSOUT=*
//SLSIN        DD   *
 CONFIG RESET CDSLEVEL(V62ABOVE)
 GLOBAL MAXVTV=65000 MVCFREE=60 VTVATTR=SCRATCH RECALWER=YES
 LOCKSTR=STK_VTCS_LOCKS VTVPAGE=LARGE
 RECLAIM THRESHLD=70 MAXMVC=30 START=98 CONMVC=1
TAPEPLEX THISPLEX=TAPEPLXB RECVPLEX=TAPEPLXA
VTSS NAME=VTSSB LOW=75 HIGH=80 MAXMIG=8 MINMIG=1 RETAIN=10
 RTD  NAME=VSMB3A00 DEVNO=3A00 CHANIF=0C
 RTD  NAME=VSMB3A01 DEVNO=3A01 CHANIF=0D
 RTD  NAME=VSMB3A02 DEVNO=3A02 CHANIF=0K
 RTD  NAME=VSMB3A03 DEVNO=3A03 CHANIF=0L
 RTD  NAME=VSMB4A08 DEVNO=4A08 CHANIF=1C
 RTD  NAME=VSMB4A09 DEVNO=4A09 CHANIF=1D
 RTD  NAME=VSMB4A0A DEVNO=4A0A CHANIF=1K
 RTD  NAME=VSMB4A0B DEVNO=4A0B CHANIF=1L
```

FIGURE 4-5 CONFIG for CTR Example - TapePlex TAPEPLXB

Defining Policies for CTR

Policies for the Sending TapePlex

To define policies for the sending TapePlex (TAPEPLXA) of the example CTR system shown in FIGURE 4-3 on page 53**, do the following:**

1. **Create MVC** POOLPARM/VOLPARM **definitions for TAPEPLXA:**

```
POOLPARM TYPE(MVC) NAME(MVCPLA) INITMVC(YES) MVCFREE(25) -
 MAXMVC(98) THRESH(85) START(98)
VOLPARM VOLSER(AM1000-AM1299) MEDIA(STK1R)
```

FIGURE 4-6 TAPEPLXA POOLPARM/VOLPARM Definitions

2. **Create VTV** POOLPARM/VOLPARM **scratch pool definitions for TAPEPLXA:**

```
POOLPARM TYPE(SCRATCH) NAME(ASCRPL)
VOLPARM VOLSER(AV1000-AV1999) MEDIA(VIRTUAL) REC(VIRTUAL)
```

FIGURE 4-7 TAPEPLXA VTV Scratch Pool Definitions

3. **For TAPEPLXA, create the Storage Classes for the MVCs that contain the locally migrated VTVs and the CTR Storage Classes.**

```
STOR NAME(LOCAL1) ACS(00) MEDIA(STK1R)
STOR NAME(EEPA1) TAPEPLEX(TAPEPLXB)
```

FIGURE 4-8 TAPEPLXA Storage Classes

In FIGURE 4-8, the STORclas statements define:

- Storage Class LOCAL1, which is Storage Class for the locally migrated VTVs from each VTSS.
- Storage Class EEPA1, which is a Storage Class for CTR, and specifies the receiving TapePlex (TAPEPLXB).

4. **Create the Management Class that points to the Storage Classes in** Step 3**.**

```
MGMT NAME(LOCEEX1) MIGPOL(LOCAL1) EEXPOL(EEPA1)
```

FIGURE 4-9 Management Class for Replication/CTR

5. **Create an SMC Policy that specifies virtual media and assigns the Management Class created in** Step 4**.**

```
POLICY NAME(PPAY) MEDIA(VIRTUAL) MGMT(LOCEEX1)
```

6. **Create a TAPEREQ statement to route critical data to VSM and assign the corresponding Policy to the data.**

```
TAPEREQ DSN(*.PAYROLL.**) POLICY(PPAY)
```

FIGURE 4-10 TAPEREQ Statement to Route Data, Assign Policy

In FIGURE 4-10, the TAPEREQ statement specifies to route data sets with HLQ mask *.PAYROLL.** to VSM and assign Policy PPAY.

Note – Also note the following:

- Although you can use SMC policies to direct your CTRs to a specific esoteric, StorageTek recommends using **only** MGMTCLAS so that the SMC/VTCS allocation influencing can use any VTSS that supports the MGMTCLAS requirements.
- You can use the EEXPORT command to do manual CTR. For more information, see *ELS Command, Control Statement, and Utility Reference*.

7. **Check your SYS1.PARMLIB SMFPRMxx member to ensure that subtype 28 records are enabled.**

   If enabled, VTSS writes a subtype 28 record that includes the target VTSS name for each CTR event. And speaking of VTSSs, I'll know that my CTR has succeeded when the VTVs arrive successfully at receiving site's VTSS...is there any way I can find that out? Yes, via the DRMONitr utility, as described in Step 8.

8. **Create JCL to monitor the CTR.**

   For this, we use the DRMONitr utility to monitor the CTR. DRMONitr causes the associated MVS job to pause until the CTR completes successfully. For example:

```
//MONITOR EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//* If HSC IS NOT OR MAY NOT BE ACTIVE, INCLUDE THE
//* FOLLOWING:
//SLSCNTL DD DSN=primary.cds.name,DISP=SHR
//SLSCNTL2 DD DSN=secondary.cds.name,DISP=SHR
//SLSSTBY   DD DSN=standby.cds.name,DISP=SHR
//SLSPARMP DD DSN=hlq.PARMLIB(BKPCNTL),DISP=SHR
//SLSPARMS DD DSN=hlq.PARMLIB(BKPCNTL2),DISP=SHR
//SLSPARMB DD DSN=hlq.PARMLIB(BKPSTBY),DISP=SHR
//SYSIN         DD UNIT=SYSDA,SPACE=(TRK,1)
//* THE FOLLOWING IS USED BY THE SNAPSHOT UTILITY:
//SYSPRINT  DD  SYSOUT=*
//SLSPRINT   DD SYSOUT=*
//SLSIN        DD *
DRMON MGMT(LOCEEX1) STOR(EEPA1) MAXAGE(24) TIMEOUT(120)
```

   In this example, the DRMON utility monitors the migrates for Storage Class EEPA1, which is the CTR Storage Class specified by Management Class LOCEEX1 in Step 4. Additionally, monitor only VTVs that have been updated in the last 24 hours, and time out DRMON after 120 minutes.

Policies for the Receiving TapePlex

To define policies for the receiving TapePlex (TAPEPLXB) of the example CTR system shown in FIGURE 4-3 on page 53**, do the following:**

1. **Create MVC** POOLPARM/VOLPARM **definitions for MVC Pool defined for TapePlex TAPEPLXB to hold CTR VTVs from TAPEPLXA:**

```
POOLPARM TYPE(MVC) NAME(AMVCCTR) INITMVC(YES) MVCFREE(25) -
MAXMVC(98) THRESH(85) START(98)
VOLPARM VOLSER(BM1000-BM1099) MEDIA(STK1R)
```

FIGURE 4-11 MVC Pool defined for TapePlex TAPEPLXB to hold CTR VTVs from TAPEPLXA

Note – StorageTek strongly recommends that you use the POOLPARM/VOLPARM feature to ensure that volume ranges are reserved for CTR-replicated volumes at the remote site.

---

2. **Create an External VTV Pool for TAPEPLXA exported VTVs:**

```
POOLPARM TYPE(EXTERNAL) NAME(AEXTBPL) OWNRPLEX(TAPEPLXA)
VOLPARM VOLSER(AV1000-AV1999) MEDIA(VIRTUAL) REC(VIRTUAL)
```

FIGURE 4-12 External VTV Pool defined for TAPEPLXA exported VTVs

---

Note – FIGURE 4-12 does not define any pools for production work on TAPEPLXB, only pools used by TAPEPLXA. If production work is to be run on TAPEPLXB, then additional POOLPARM and VOLPARM definitions are needed for the scratch and MVC pools for TAPEPLXB work.

---

3. **Create a VTV Scratch Pool for TapePlex TAPEPLXB use for TAPEPLXA work:**

```
POOLPARM TYPE(SCRATCH) NAME(ASCRPL)
VOLPARM VOLSER(BV1000-BV1999) MEDIA(VIRTUAL) REC(VIRTUAL)
```

FIGURE 4-13 VTV Scratch Pool defined for TapePlex TAPEPLXB use for TAPEPLXA work:

4. **Create an MVC Pool for TapePlex TAPEPLXB to hold VTVs from TAPEPLXA DR test or production (in case of a disaster):**

```
POOLPARM TYPE(MVC) NAME(AMVCDR) INITMVC(YES) MVCFREE(25) -
MAXMVC(98) THRESH(85) START(98)
VOLPARM VOLSER(BM2000-BM2099) MEDIA(STK1R)
```

FIGURE 4-14 MVC Pool defined for TapePlex TAPEPLXB to hold VTVs from TAPEPLXA DR test or production

5. **For TAPEPLXB, create the Storage Classes for local migration.**

```
STOR NAME(TPEPLXA1) MVCPOOL(AMVCCTR)
STOR NAME(TPEPLXA2) MVCPOOL(AMVCDR)
```

FIGURE 4-15 Storage Classes for Local and Remote Migrated VTVs

In FIGURE 4-15, the STORclas statements define Storage Classes TPEPLXA1 and TPEPLXA2 for local migration. The Storage Class names allow us to segregate this work from the TAPEPLXB local work.

6. **Create the Management Classes that point to the Storage Classes in** Step 5**.**

```
MGMT NAME(LOCEEX1) MIGPOL(TPEPLXA1)
MGMT NAME(LOCPLXA) MIGPOL(TPEPLXA2)
```

FIGURE 4-16 Management Classes for Replication

Note that the name LOCEEX1 matches Management Class name that we used on TAPEPLXA (this Management Class is specified in the VTV metadata that is sent from the VTSS on TAPEPLXA), but we reference the Storage Class for local migration. The definitions of the Management and Storage Classes on TAPEPLXB can use any parameters including EEXPOL to replicate to a third TapePlex. In addition, we create another MGMTCLAS, LOCPLXA, to be used for migration during a DR test of TAPEPLXA's workload.

Using CTR When the Remote Site Has No LPARs

In some environments only one site has LPARs doing tape activity, while a second site contains only library and VTSS hardware but no MVS LPARs. It is possible to set up this environment so that CTR can be used as a DR and DR test mechanism.

To do this, you must:

1. **Run the SMC client/server feature on your production environment so that you have at least one production LPAR that is not running HSC/VTCS.**

   Alternatively, you can run the DR TapePlex on the same LPAR as the production TapePlex using the MULT mode feature. See *Configuring HSC and VTCS* for more information on using this capability.

   The production TapePlex in this example is TAPEPLXA.

2. **Create a new CDS defining the hardware (libraries and VTSSs) in your remote site.**

3. **Start an HSC/VTCS using the new CDS on an LPAR (MVSX) that is not currently running production HSC/VTCS., or on the LPAR where you have decided to run multiple copies of HSC/VTCS using the MULT mode feature.**

   Note – For reliability, it is recommended that you may want to run two instances of HSC/VTCS pointing to TAPEPLXB on two different LPARs, so that if one instance is unavailable, metadata for the Cross-TapePlex replicated VTVs can be sent to the second instance.

   This system is TapePlex TAPEPLXB.

4. **Define parameters for the SMC system on MVSX defining both the TAPEPLXA and TAPEPLXB TapePlexes.**

   Each SMC system in the complex must define both TapePlex TAPEPLXA (the production TapePlex) as well as TAPEPLXB, the DR TapePlex. In order to support continuing to replicate VTVs during a DR test, you must define a server for TapePlex TAPEPLXB that points to the host at the remote site. For example:

```
TAPEPLEX NAME(TAPEPLXA) LOCSUB(HSCA)
TAPEPLEX NAME(TAPEPLXB)
SERVER NAME(TPLXBPR) TAPEPLEX(TAPEPLXB) HOST(MVSX) PORT(999)
SERVER NAME(TPLXBDR) TAPEPLEX(TAPEPLXB) HOST(MVSXDR) PORT(1234)
```

   FIGURE 4-17 TapePlex/Server Definitions - No LPARs at Remote Site

   ---

   Note – This example assumes that although the LPAR names (MVSX) may be identical between the production and DR site, the two sites have unique TCP/IP host names.

   ---

5. **Define your VTCS policies on TAPEPLXA to allow CTR to TAPEPLXB.**

   See "Defining Policies for CTR" on page 56.

6. **Using your disk replication solution, maintain a copy of the contents of the CDS for TAPEPLXB at your remote location.**

   Alternatively, if reliable connectivity exists, you may want to maintain the primary (and other) copies of the HSC CDS at the DR site, using FICON connections to access the CDS from the production site.

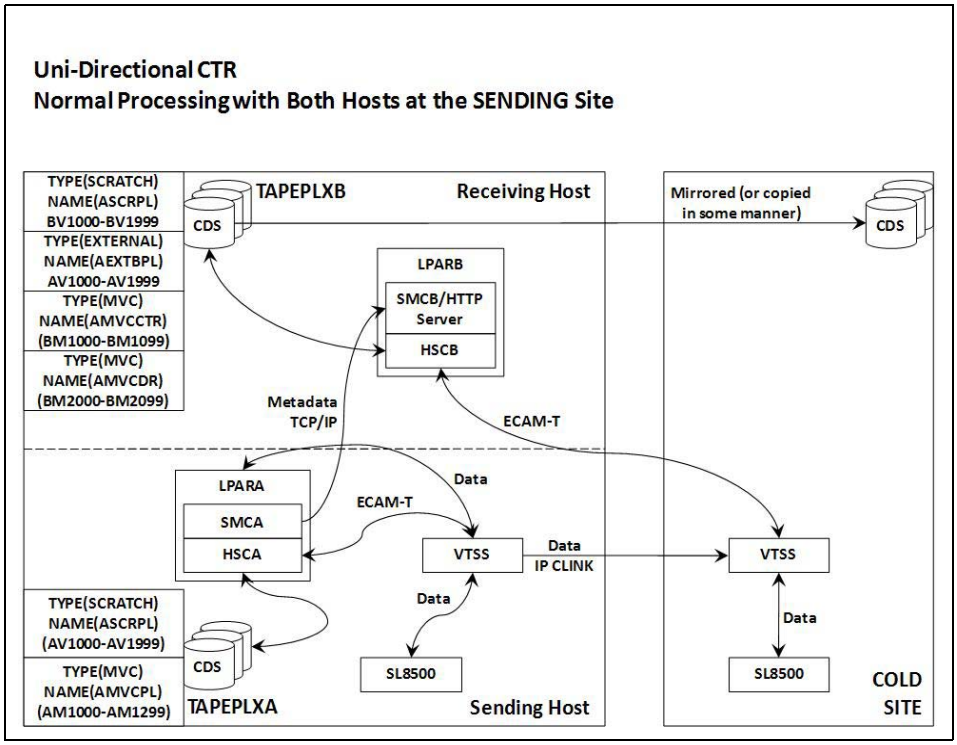

FIGURE 4-18 CDS Copy - No LPARs at Remote Site

Using CTR as a DR Solution

A DR solution always lets you do three things:

- **Set up and start the solution** as described in "The Setup: Configuring and Starting CTR" on page 54.
- If a disaster occurs, **use the solution to continue doing business** at the remote site, as described in "Using CTR for Business Continuance" on page 63.
- **Use the solution to resume doing business** at the local site after it is up and running again as described in "Using CTR for Business Resumption" on page 65.

Using CTR for Business Continuance

If site TAPEPLXA has an outage, you can to continue doing business at site TAPEPLXB simply by running the workload using TAPEPLXB's TapePlex. In order to protect the data, the VTVs that were replicated from the TAPEPLXA remain in a read-only state (see Section CTR VTV Read-Only Considerations). However, once you have successfully re-established the TAPEPLXA workload, you may want to scratch some of the VTVs that were replicated from TAPEPLXA. Note that you should be careful before you perform this step to ensure that your production work for both TAPEPLXA and TAPEPLXB is stable. Also, at some future point you will probably want to re-create a separate TapePlex environment for TAPEPLXA to return to your original configuration.

To use CTR for Business Continuance:

1. **1.Change the POOLPARM/VOLPARM definitions in the TAPEPLXB TapePlex CDS for the pool named (AEXTBPL) from TYPE(EXTERNAL) to TYPE(SCRATCH):**

```
POOLPARM TYPE(SCRATCH) NAME(AEXTBPL)
VOLPARM VOLSER(AV1000-AV1999) MEDIA(VIRTUAL)
```

Note that the VOLPARM VOLSER range remains unchanged.

1. **You can now run a scratch synchronization job under TAPEPLXB to scratch VTVs in the AV1000-AV1999 range based on the scratch status in the TMS, either to return to a checkpoint or to perform normal scratch update processing.**

You may want to allow some time to pass between starting production processing and allowing the VTV volume serial numbers in the range AV1000-AV1999 to be re-used as scratch volumes. The use of the POOLPARM/VOLPARM feature ensures that these volumes cannot be selected as scratch unless a policy specifically requests SUBPOOL(AEXTBPL).

During this period you will use volumes in the TAPEPLXB scratch subpool ASCRPL (volser range BV1000-BV1999) for TAPEPLXA production work.

Once your disaster recovery environment is stabilized, you can again change your POOLPARM/VOLPARM definitions to allow scratch volumes to be selected from the AV1000-AV1999 range:

```
POOLPARM TYPE(SCRATCH) NAME(ASCRPL)
VOLPARM VOLSER(AV1000-AV1999) MEDIA(VIRTUAL)
VOLPARM VOLSER(BV1000-BV1999) MEDIA(VIRTUAL)
```

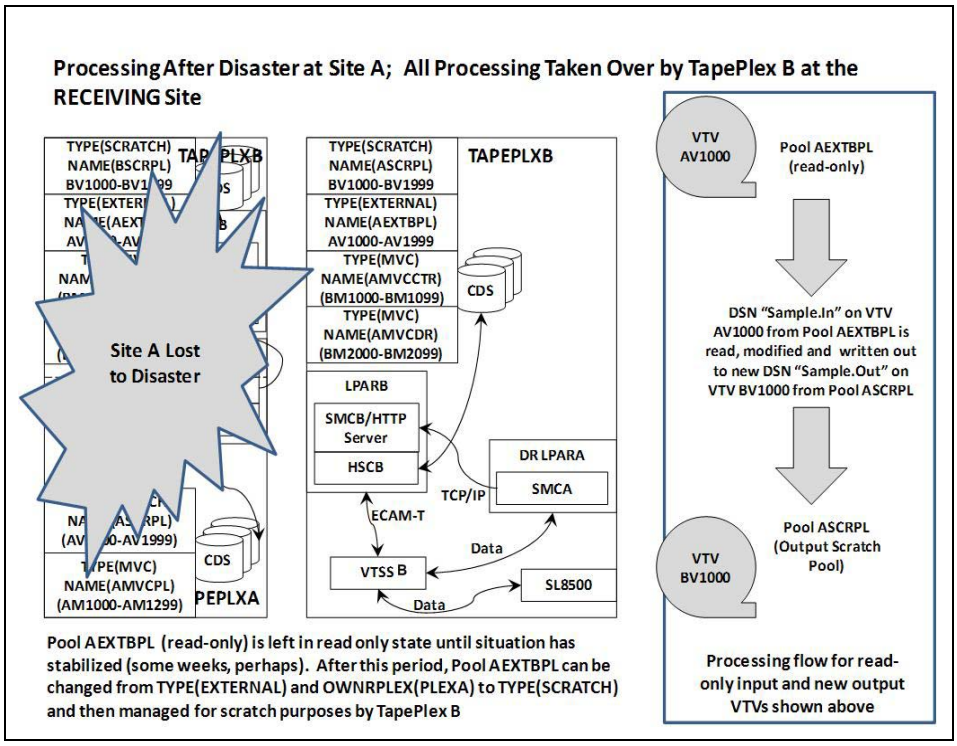

FIGURE 4-19 System During Business Continuance

Using CTR for Business Resumption

The local site experienced an outage, we continued doing business at the remote site, now the local site is up and running again, so how do I resume doing business at the local site? Essentially, my business resumption depends on what happened during and following the outage. Assume that all of your original local data was lost, and you have a brand-new empty VTSS at the local site.

To resume business after losing all data at the local site:

1. **Create a new CDS and run an HSC audit to determine the contents of the physical libraries.**

   You then need to "reverse replicate" the data and the metadata to the local site from the remote site.

1. **Set up your CONFIG deck for the remote site so it can send data to the local site.**
2. **Reverse replicate using EEXPORT.**

   For example:

   ```
   EEXPORT MGMTCLAS(LOCEEX1,LOCEEX2) TOPLEX(TAPEPLXA)
   ```

Disaster Recovery Testing Using Cross-Tapeplex Replication

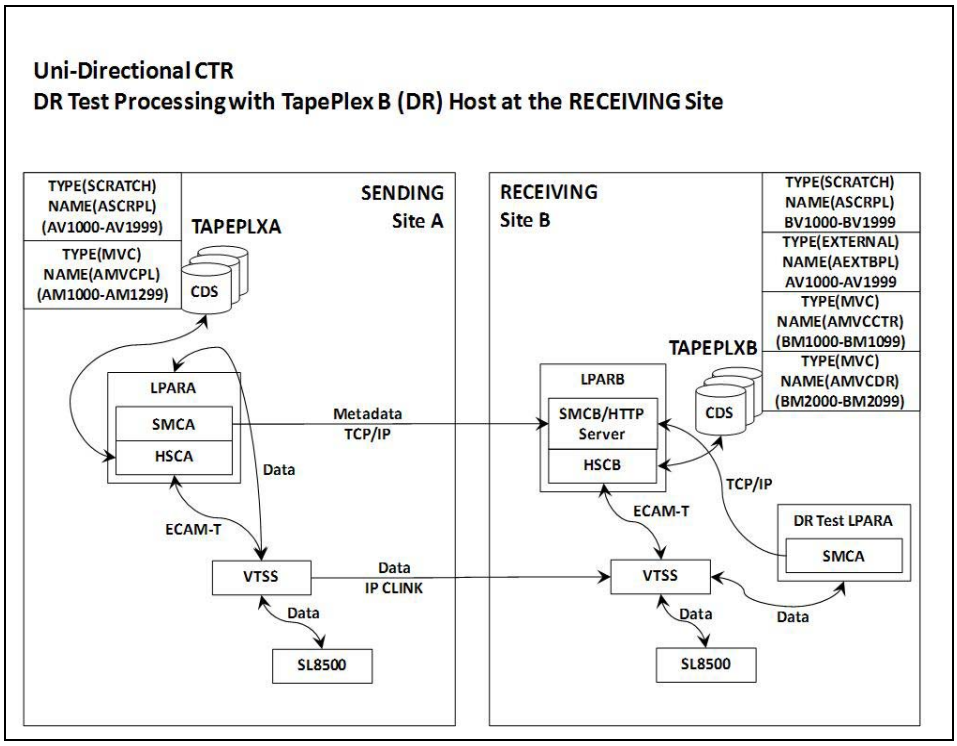

FIGURE 4-20 Disaster Recovery Testing Using Cross-Tapeplex Replication

To continue with our example, we have two sites, TAPEPLXA and TAPEPLXB, each defined as its own TapePlex (HSC CDS), and that you have used the Cross-TapePlex Replication feature to replicate your critical VTVs from TAPEPLXA to TAPEPLXB.

To perform a DR test at TAPEPLXB for TAPEPLXA's work, the following procedure is recommended:

1. **Ensure that the TAPEPLXB CDS contains one or more scratch subpools for TAPEPLXA output data which are separate from scratch subpools used for TAPEPLXB work.**

   For examples, see "Policies for the Receiving TapePlex" on page 58.

2. **Ensure that catalog and tape management data from TAPEPLXA is available.**

3. **Bring up SMC on your TAPEPLXA test LPAR, defining its TapePlex as TAPEPLXB, specifying SERVER commands for one or more HSC hosts in TAPEPLXB.**

4. **Begin executing your test workload.**

   Your SMC will automatically have access to VTVs that existed either before or after the start of the test, because these are continuing to be replicated from TAPEPLXA. Ensure that the VTVs that your DR test will use are not scratched or altered by the TAPEPLXA TapePlex.

5. **When the test is complete, scratch all VTVs in the DR test subpool(s) used by the test.**

   When using this approach, no special CDS is required, and no special rules are needed to ensure that two separate HSC systems can share hardware resources. However, this method **requires** that the DR test be executed using current data, or at least data that is currently available. Note that your DR test output as well as your TAPEPLXA replicated VTVs will use TAPEPLXB VTSS buffer space.

   Because the data that was replicated from the TAPEPLXA TapePlex is read-only, any attempts by the DR test to modify the data will result in message SMC0247, Mount failed for write-protected VTV vvvvvv on drive dddd from SMC indicating that the VTV cannot be mounted. The occurrence of this message may indicate that your DR process does not have clearly defined application checkpoints (see "CTR VTV Read-Only Considerations" on page 52). If this is the case, the use of CTR for your DR strategy may not be a good choice.

Note – Note that if you use Cross-TapePlex replication to create copies of your VTVs at a remote site, it is recommended that you do not use CDRT for your DR testing, as the use of CDRT will not permit the read-only VTVs to be updated, even in a separate CDRT environment.

DR Testing When the DR Site Has No LPARs

When you manage the CTR hardware at the DR site using a TapePlex executing at the production site, there are a few additional considerations for a DR test. This example uses TAPEPLXA for the production TapePlex and TAPEPLXB for the TapePlex that normally runs at the production site but runs at the DR site during a DR test.

1. **You must stop the DR TapePlex TAPEPLXB at the production site prior to the test.**

   During the DR test, the TAPEPLXB will be executing at the DR site on a copy of the TAPEPLXB CDS.

2. **Production VTVs will continue to be sent to TAPEPLXB and reflected in the CDS at the DR site.**

   During this time the TAPEPLXB CDS at the production site becomes outdated, as it no longer reflects VTVs that are being replicated during the DR test. The TAPEPLEX and SERVER statements on the production LPARs ensure that data replication continues during the DR test:

   ```
   TAPEPLEX NAME(TAPEPLXB)
   SERVER NAME(TPLXBPR) TAPEPLEX(TAPEPLXB) HOSTNAME(MVSX)
   PORT(999)
   SERVER NAME(TPLXBDR) TAPEPLEX(TAPEPLXB) HOST(MVSXDR) PORT(1234)
   ```

3. **When you start the HSC/VTCS for TapePlex TAPEPLXB at the DR site, you must be sure to start the HTTP server on SMC:**

   ```
   HTTP START PORT(1234)
   ```

   The port number (1234) matches what was defined in the TAPEPLXBDR SERVER statement.

4. **At the conclusion of the test, scratch all VTVs that were created by the test.**

   You do not have to determine what VTVs were actually created by the test; you can simply scratch all volumes in the subpool(s). For example:

   ```
   SCRATCH VOL(BV1000-BV2999)
   ```

5. **Stop the HSC/VTCS for TAPEPLXB at the DR site.**

6. **You must now ensure that the TAPEPLXB CDS at the DR site is sent back to the production site.**

   Ideally this can be done by mirroring the TAPEPLXB CDS back to the production site during the DR test. If this is not possible, you can use FTP or other mechanism of your choice to copy the CDS from the current version at the DR site back to the production site.

7. **Restart TAPEPLXB on the LPAR(s) at the production site.**

   While there is no active copy of TAPEPLXB, VTVs scheduled for CTR to TAPEPLXB will remain in the VTSS buffer. When TAPEPLXB is active again at the production site, these VTVs will be replicated to the VTSS at the DR site.

Managing VTVs Replicated via Cross-TapePlex Replication (CTR)

You can use VTVMAINT to change the status of VTVs replicated via CTR as follows:

- Use VTVMAINT DELEXpot to remove the name of a TapePlex that references a VTV. For example, if you replicate a VTV from TAPEPLXA to TAPEPLXB, then delete the copy on TAPEPLXA, you can use VTVMAINT DELEXpot to remove TAPEPLXA's reference to the VTV.
- Use VTVMAINT ADDEXpot to add the name of a TapePlex that references a VTV as described in "Using CTR for Business Continuance" on page 63.
- The VTVMAINT utility can be used to change the ownership of a VTV that was received via CTR, but the VTV must be currently in scratch status. For example, VTVMAINT OWNRPLEX(TAPEPLXB) will change the ownership of a VTV sent from TAPEPLXA to be owned by the TapePlex where it currently resides.

CHAPTER 5

Configuring a Remote Library

FIGURE 5-1 shows an example configuration with a remote library; the following sections tell how to configure this example.

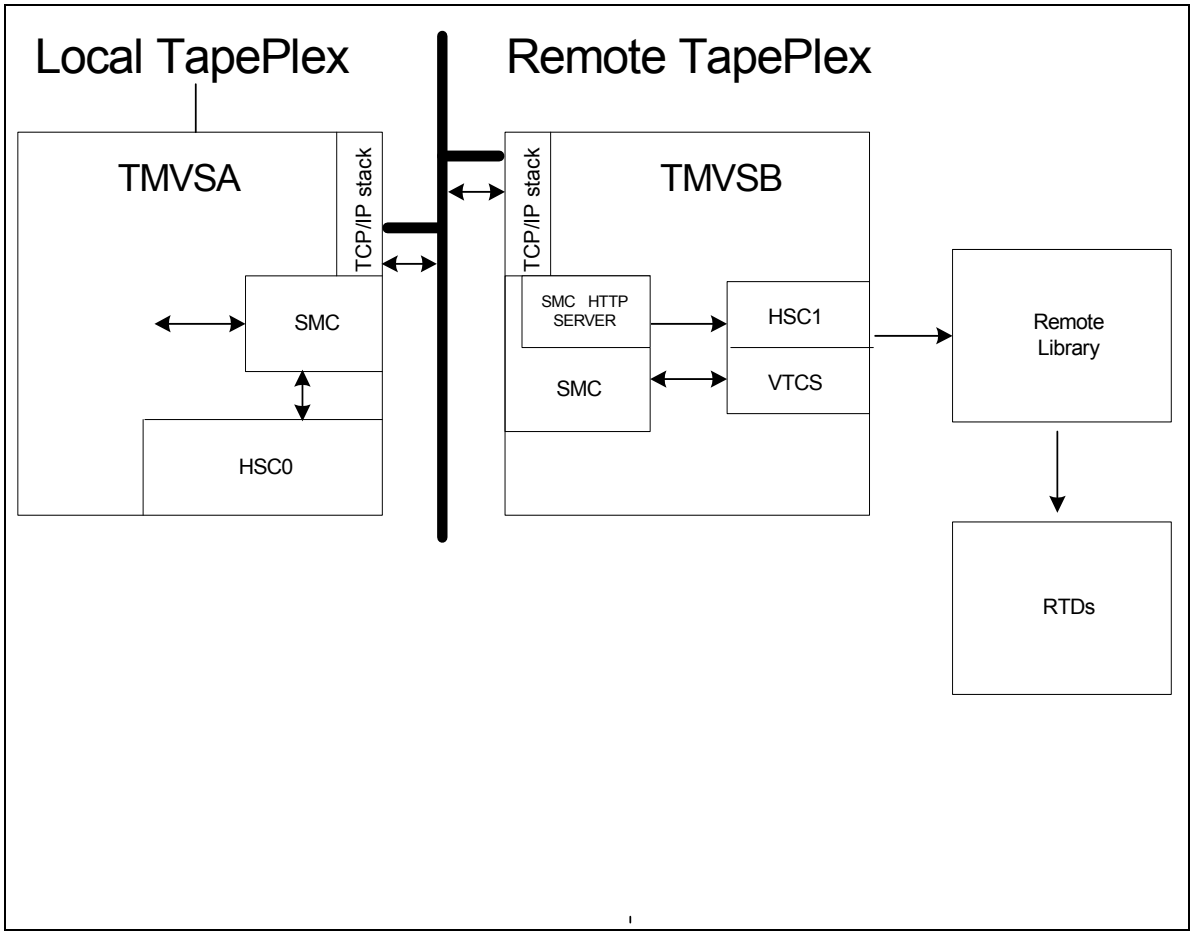

FIGURE 5-1 RTDs Operating Under a Remote TapePlex

Modifying the SMC SCMDS file

SMC manages all communication between VTCS and a remote TapePlex, so SMC must know how to connect to the remote TapePlex. You do so by adding an SMC STORMNGR statement for each remote TapePlex plus one or more SMC SERVER statements that define the TCP/IP control paths to the remote TapePlex. You may want to do this in your SMC CMDS file as shown in CODE EXAMPLE 5-1.

```
TAPEPLEX NAME(TMVSA)LOCSUB(HSC0)
TAPEPLEX NAME(TMVSB)LOCSUB(HSC1)
SERVER NAME(TMVSB)IP(192.168.1.10)PORT(60000)
```

CODE EXAMPLE 5-1 SMC Commands for a Remote Library

CODE EXAMPLE 5-1 contains:

- A TAPEPLEX statement, which defines a local TapePlex, TMVSA, with an HSC running on the local MVS host (HSC0).
- A second TAPEPLEX statement, which defines a remote TapePlex, TMVSB, with an HSC running on the remote MVS host (HSC1).
- A SERVER command that defines a UUI communication path to TMVSB, where:
  - The remote server name is TMVSB.
  - The IP parameter value is the ELS port IP address of 192.168.1.10 for UUI communications.
  - The PORT parameter value is 60000; this value is always used for the SERVER PORT parameter for SMC communication with a TMVSB.

Updating the VTCS CONFIG Deck to Define a Remote Library

You must update VTCS `CONFIG` deck to define the remote library and the connectivity from the VTSS to the remote library. As shown in CODE EXAMPLE 5-2, the remote library is defined via a `CONFIG STORMNGR` statement.

```
TAPEPLEX THISPLEX=TMVSA
STORMNGR NAME=TMVSB
VTSS NAME=VTSS1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTDPATH  NAME=RM1RTD1 STORMNGR=TMVSB DEVNO=1A00 CHANIF=0A:0
RTDPATH  NAME=RM1RTD2 STORMNGR=TMVSB DEVNO=1A01 CHANIF=0A:1
RTDPATH  NAME=RM1RTD3 STORMNGR=TMVSB DEVNO=1I00 CHANIF=0I:0
RTDPATH  NAME=RM1RTD4 STORMNGR=TMVSB DEVNO=1I01 CHANIF=0I:1
VTD LOW=6900 HIGH=69FF
```

CODE EXAMPLE 5-2 **VTCS CONFIG Remote Library Example**

In CODE EXAMPLE 5-2 on page 73, note:

- The `CONFIG TAPEPLEX` statement specifies `TMVSA` as the local TapePlex.
- A `STORMNGR` statement specifies `TMVSB` as the remote library.
- The `CONFIG RTDPATH` statements for VTSS1, which specify:
  - The name of the `RTDPATH`.
  - The connection to the remote library (`STORMNGR=TMVSB`).
  - The device number (DEVNO).
  - The `CHANIF` value for each VTSS to RTD connection in *ci:p* format where:
    - *c* is 0 or 1.
    - *i* is A or I.
    - *p* is 0 through 3.

    Note – For VSM5s, the CHANIF value must match the values specified on the VSM5 `IFF Configuration Status Screen`. For VSM 6s, this must be unique for each VTSS but does **not** correspond to an actual value on the VSM 6 TCP/IP ports.

You can now use the STORclas STORMNGR parameter to route data to the remote library. For example:

```
STOR NAME(REMLIB) STORMNGR(TMVSB)
```

MVC Pool Considerations

Any MVCs in a remote library must be included in the local MVC pool definitions, preferably via VOLPARM/POOLPARM definitions. The local HSC treats the remote MVCs as non-library.

Any remote library server typically needs to define the MVCs as "externally managed" by some method. Having a remote TapePlex sharing a pool of local MVCs is **not** supported because there is no serialization.

If a pool of MVCs is shared for data transfer purposes, then only one TapePlex can actively write to the MVCs. The other TapePlexes can have only read-only access. Serialization of the MVCs on drives is the responsibility of the user.

CHAPTER 6

Using Clustered VTSS Configurations

Ever wish you could copy VTVs from one VTSS to another? Well, you can, thanks to the magic of Clustered VTSSs. Clustered VTSS is a powerful tool for applications such as but not limited to DR (Disaster Recovery) solutions. As you've probably guessed, however, with Clustered configurations, Some Assembly *is* Required. So let's start with the basics of what VTSS Clusters are and how they work:

After the basics, there are additional variations and features of Clustered VTSS that you'll want to know about:

What is Clustered VTSS?

A VTSS cluster is a High Availability (HA) solution providing for maximum data availability. It consists of two or more VTSS systems connected via FICON or TCP/IP communication links (CLINKs). Additionally, every VTSS system within the cluster can access all data created within the cluster (VTSS resident or migrated). Data (VTVs) created on a cluster are replicated from one VTSS system to another VTSS within the same cluster under the control of VTCS policies.

Note – To ensure that every VTSS system can access all data created within the cluster, clustered configurations can be either of the following:

- Each VTSS in the cluster has attached either RTDs or VLEs.
- "Tapeless" - no VTSSs have VLEs or RTDs attached.

Clustered configurations, therefore, provides the highest data availability with a hot recovery if a VTSS within a cluster has an outage (replicated data remains available without requiring recalls from MVCs).

Prior to VTCS 7.0, a cluster could only consist of two VTSSs. With VTCS 7.0, many VTSSs can form a single cluster. A VTV, however, can only be resident in two VTSSs at any point in time.

A cluster can span geographic locations. A cluster, however, **must be within a single TapePlex** (controlled by a single CDS).

A VTV can be replicated (copied) from one VTSS to another either:

- Asynchronously to the VTV creation – scheduled to complete as soon as possible after the VTV dismount
- Synchronously with the VTV creation. The VTV dismount will not complete until the replication is complete.

Note – The VTSS estimates if the VTV can be synchronously replicated within 40 minutes. If this is not possible, the VTV is asynchronously replicated.

The connections between VTSS systems within a cluster can be either uni-directional, where data (VTVs) flows only one way or bi-directional, where data (VTVs) can flow in both directions. The CONFIG utility specifies whether a cluster is uni- or bi-directional, and a VTVs Management Class determines its replication policy, if any, and whether the replication is done synchronously or asynchronously.

So both vaulting MVCs (as described in "Using the ELS External Vaulting Feature" on page 29) and VTV replication can facilitate a Disaster Recovery/Business Continuance solution. VTV replication, however, is superior as a High Availability solution because with replication:

- Data can be backed up synchronously.
- Recent data, which has been replicated to a "Recovery" VTSS, can be restored more quickly because you don't have to mounting MVCs.

Clustered VTSS Requirements

TABLE 6-1 Clustered VTSS Requirements

Component	Requirement
Extended Clusters	D02.07.00.00 or greater VTSS microcode for VSM4s or VSM5s with FICON connections only. For VSM 6s, all levels of microcode.
2 VTSSs within a cluster (ESCON Interfaces)	The Primary and Secondary VTSSs can be any combination of VSM4s where the Secondary can be of any capacity. VSM5s do not have ESCON interfaces, and cannot be in a cluster with other VTSSs that use ESCON.
2 VTSSs within a cluster (FICON Interfaces)	The Primary and Secondary VTSSs can be any combination of VSM4 and VSM5 where the Secondary can be of any capacity. For example, all of the following are valid: ■ Primary VSM5, Secondary VSM4 ■ Primary VSM5, Secondary VSM5 ■ Primary VSM4, Secondary VSM4 ■ Primary VSM4, Secondary VSM5 (not recommended)
Primary and Secondary VTSS microcode	The Primary VTSS microcode must be at a level that supports sending replicated VTVS. The Secondary VTSS microcode must be at a level that supports receiving replicated VTVS and supports the use of the Secondary as a production VTSS. After the microcode is installed, the Clustering feature must be enabled at **both** the Primary and Secondary VTSS via an options floppy disk. See your StorageTek hardware service representative for details.

TABLE 6-1 Clustered VTSS Requirements

VTDs reserved for clustering	In Clustered VTSS configurations, you must ensure that the first 16 VTDs in each VTSS (0-F) are reserved for clustering. These devices must be OFFLINE to MVS, and their paths must be online to each HSC server host. This also applies to any VTSSs involved in Cross TapePlex Replication. VTCS does not register the first 16 VTDs with SMC/HSC, which prevents mounting VTVs on these VTDs.
RTDs	In dual-ACS environments, the same device types must be represented in the RTDs attached to each ACS so that data migrated by one VTSS can be recalled by the other VTSS. The number of MVCs, the media type and location used for the migration is determined by the MIGPOL parameter of the MGMTclas statement. Each ACS needs to be considered and if a drive type in one ACS is connected to one of the VTSSs in a clustered VTSS environment, a drive of the same type and in the same ACS needs to be connected to every other VTSS in that clustered environment.
Native IP (clustering with TCP/IP)	Native IP requires CDSLEVEL F and above is required, with the following PTFs: ■ For 6.2: ■ L1A00P7 - SMC6200 ■ L1H14IM - SMS6200 ■ L1H14O2 - SOS6200 ■ L1H14IL - SWS6200 ■ For 7.0, L1H150G (SES7000) ■ For 7.1 and above, support is included in the base. The following connections are supported for Native IP: ■ VSM5 to VSM5 ■ VSM5 to VSM 6 ■ VSM 6 to VSM 6 ■ VSM 6 or VSM5 to VLE

Synchronous replication, which applies to only VSM4s and above, has the requirements described in TABLE 6-2.

TABLE 6-2 Synchronous Replication Requirements

Synchronous replication requires...	**..the following VTSS microcode...**	**...and CDS level...**
Native IP or FICON ports for the CLINKs	D02.03.00.00 or higher for VSM4s and VSM5s, for VSM 6s, all levels of microcode	"F" or higher

How Clustered VTSS Configurations Work

You can use VSM to connect two VTSSs by Cluster Links (CLINKs) to form a *Clustered VTSS configuration*. You use the following statements to implement a Clustered Configuration:

- Clusters can be either Uni-Directional or Bi-Directional depending on the CLINK statements.
- The Secondary VTSS (or the second Peer) can either be at the same physical location as the Primary (or first Peer) or at a remote location.
- The CONFIG CLUSTER statement specifies the VTSSs that form the Cluster.
- The CONFIG CLINK statement defines the CLINKs that connect the VTSSs. The way you write the CLINK statements determines whether the replication is uni-directional or bi-directional. For examples, see FIGURE 6-2 on page 85 and FIGURE 6-4 on page 87.
- The MGMTclas REPLICAT parameter identifies the Management Class that contains the VTVs that VSM *replicates* (copies) from one VTSS in the Cluster to the other.

Note – The CONFIG GLOBAL REPLicat parameter now specifies when to replicate a VTV as follows:

REPLicat

specifies when VSM replicates the VTV.

ALWAYS

The replicate request is added to the VTCS replication queue every time the VTV is dismounted, regardless of whether the VTV was changed while it was mounted (the default).

CHANGED

The replicate request is added to the VTCS replication queue if the VTV:

•Was changed while it was mounted **or**

•Was only read while mounted but less than the expected number of MVC copies of the VTV exist.

Regardless of the CONFIG GLOBAL REPLicat setting, replication **also** requires that:

- The VTV must be dismounted in a VTSS that supports replication **and** there cannot be an identical copy of the VTV in the other VTSS in the Cluster.
- In addition to the CONFIG GLOBAL REPLicat value, you **must** specify REPLICAT(YES) on a VTV's Management Class for replication to occur.

For more information, see *ELS Command, Control Statement, and Utility Reference*.

- VTCS immediately migrates (with KEEP) replicated VTVs. You can specify the source VTSS for migration of replicated VTVs on the MIGRATE parameter of the STORclas statement. **Also note** that you **must** specify replication on a Management Class **that points** to a Storage Class **with** a MIGRATE parameter value to migrate from the desired VTSS. Otherwise, migration from the desired VTSS does not occur.

Because VTCS immediately migrates (with KEEP) replicated VTVs regardless of the MGMTclas IMMDELAY setting, StorageTek **strongly recommends** that you **do not** explicitly set a MGMTclas IMMDELAY policy for replicated VTVs. If you do, VTCS honors the explicit immediate migrate request, and immediately migrates the affected VTV from whichever VTSS is first capable of performing the migration (that is, the first VTSS that has a resident VTV copy and an available RTD to satisfy the migrate). Setting an explicit MGMTclas IMMDELAY policy, therefore, is redundant and may interfere with optimal VTV replication and migration.

Also note that the immediate migrate (KEEP) following replication is **not the same** as automigration. That is, during the implicit immediate migrate, no VTVs are deleted from either VTSS to manage the DBU. Instead, the VTVs are simply "pre-staged" via migration to an MVC from the receiving VTSS, leaving both VTSS buffer contents unchanged. For space management in a VTSS cluster, VTCS automigrates VTVs according to the space management/migration cycle of **either** VTSS. If the capacity of the receiving VTSS is greater than or equal to that of the sending VTSS, automigration on the sending VTSS deletes a replicate VTV from **both** the VTSSs. If the capacity of the receiving VTSS is less than that of the sending VTSS, automigration may start on the receiving VTSS. In this case, automigration deletes a replicate VTV from only the receiving VTSS, leaving the copy on the sending VTSS still resident.

- **Note that** the replication requirements of data is determined following a dismount, **not** a recall. Merely recalling a VTV will not cause a replicate – so demand recall, MVCdrain and reclaim will not cause a replicate. However, if the VTV is recalled and mounted on a VTD, at dismount time it will be replicated to the Secondary or Peer VTSS.
- A Cluster can support different workloads in each of four operating modes. For example, only Full-Function Clusters can support active replication, but in Degraded Primary Mode, you can vary the Secondary's VTDs online to MVS to take over the workload. You can use Query to display Cluster, Cluster link, VTV replication, and VTSS status. You can use VARY VTSS to change VTSS states and VARY CLink to change CLINK states.

How VTSS Reconciliation Works

- Whenever a clustered VTSS-pair resumes the full function state, VTCS reconciles the contents of the two VTSSs. This occurs either during VTCS initialization or when a VTSS goes online and its partner VTSS is also online.
- Reconciliation consists of either deleting or migrating and deleting VTVs (or replicating a VTV if this had not previously completed successfully). That is, recall is not involved in reconciling VTSS contents.

  For example, in a uni-directional cluster with a VTV resident in the receiver but not the sender VTSS, VTCS deletes the VTV from the receiver (after ensuring that all required MVC copies have been made). This avoids a recall to the sender.

  Similarly, in a uni-directional cluster with a VTV resident in the sender but not the receiver, VTSS, VTCS replicates the VTV to the receiver instead of recalling it from an MVC.
- The reconcile process assumes that if a replicated VTV is resident on the sender VTSS, then it is a valid copy. If f the copy on the receiver is different, VTCS deletes it.
- To maintain consistent reconcile actions in a bi-directional cluster, the VTSS in which the VTV is or was last resident (as indicated by the CDS VTV record), is considered to be the sender VTSS. Reconcile processing is as described above for uni-directional clusters.

Uni-Directional and Bi-Directional Clusters

Clusters of two VTSSs can be either of the following:

- **Uni-directional**, where one VTSS is the Primary and the other is the Secondary. For more information, see "Uni-Directional Clusters" on page 84.
- **Bi-directional**, where both VTSSs are peers and replication is from Peer to Peer in either direction. For more information, see "Bi-Directional Clusters" on page 86.

Uni-Directional Clusters

As shown in FIGURE 6-1, in a Uni-Directional Cluster, replication is **only** from the Primary to the Secondary.

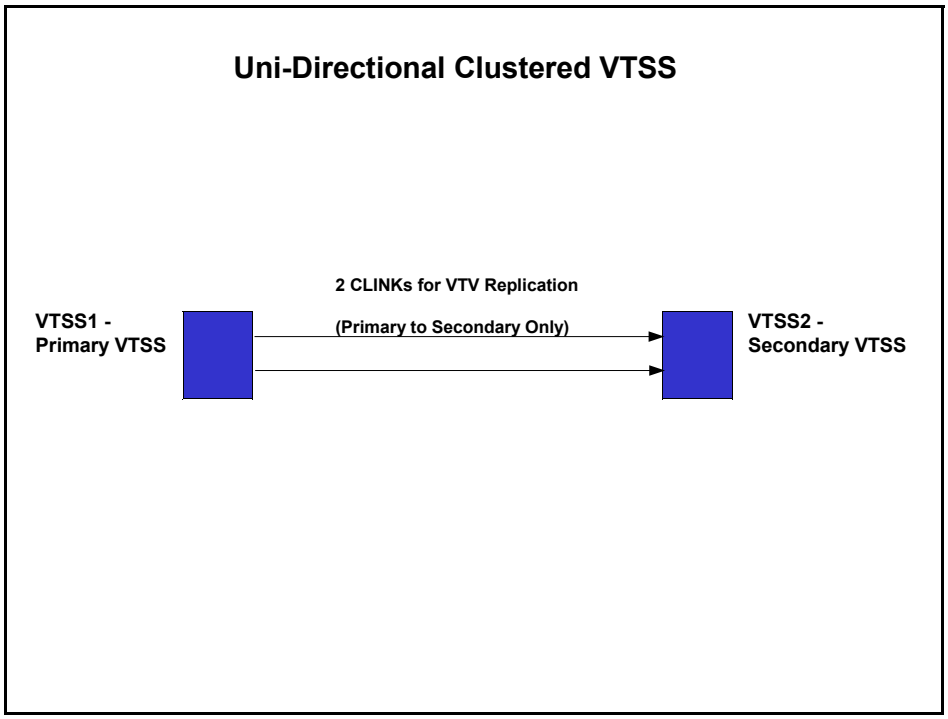

FIGURE 6-1 Uni-Directional Clustered VTSS

How Uni-Directional VTSS Clusters Work

- The Secondary can receive both replicated VTVs from the Primary and non-replicate production workload by any of the standard routing methods (for example, TAPEREQs). You need to vary the VTDs in the Secondary online to MVS so that the Secondary can accept production work. You **cannot** vary online to MVS the VTD addresses used by the CLINK terminations as described in "How Clustered VTSS Configurations Work" on page 80.
- A VTV with replication enabled is allocated to an online Primary VTSS unless none are available; in that case, the VTV is allocated to an online Secondary VTSS. If no online Secondary VTSSs are available, the VTV is allocated to a non-cluster VTSS. A VTV without replication can be allocated to any online VTSS including the Secondary of a Full-Function Cluster.
- At dismount time, a VTV with replication enabled that resides on a Full-Function Cluster is queued for replication to the Secondary VTSS. If a VTV with replication enabled is dismounted from a VTD in a VTSS that is not part of a Full-Function Cluster, the VTV is queued for immediate migration.

  When the Secondary VTSS receives a replicated VTV from the Primary VTSS, the VTV is then immediately migrated (with the KEEP option) regardless of Immediate Migrate Management Class settings for this VTV.
- **Both the Primary and the Secondary VTSS** can manage all space reclamations.
- If you are using ESCON or FICON interfaces, on the Primary VTSS, the CLINK CIPs/FIPs are configured in **Nearlink Mode**, while on the Secondary VTSS, the CIPs/FIPs are configured in **Host Mode**.

  Therefore, you configure CLINKs for **only** the Primary VTSS, as shown in the example in FIGURE 6-2, where VTSS1 is the Primary VTSS.

```
.
.
CLUSTER NAME=CLUSTER1 VTSSs(VTSS1,VTSS2)
 CLINK VTSS=VTSS1 CHANIF=0G
 CLINK VTSS=VTSS1 CHANIF=0O
 CLINK VTSS=VTSS1 CHANIF=1G
 CLINK VTSS=VTSS1 CHANIF=1O
.
.
```

FIGURE 6-2 ESCON/FICON Uni-Directional Cluster and CLINK Definitions

Bi-Directional Clusters

As shown in FIGURE 6-3, Bi-Directional Clustering, requires pairs of Uni-Directional CLINKs so that the data flows in **opposite directions** on the CLINKs.

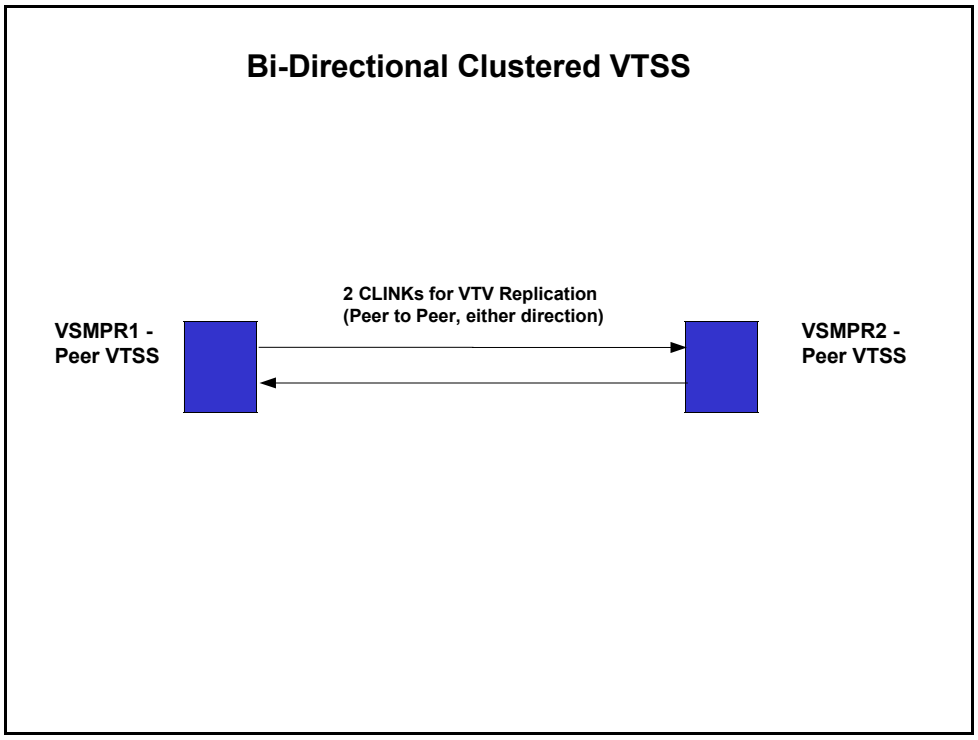

FIGURE 6-3 Bi-Directional Clustered VTSS

How Bi-Directional VTSS Clusters Work

In a Bi-Directional Cluster, in normal operation, both VTSSs are online to VTCS as follows:

- In a Bi-Directional Cluster, each of the Peer VTSSs can receive production work via the standard routing methods (for example, TAPEREQs). You need to vary the VTDs in both VTSSs online to MVS so that each can accept production work. However, **note that** you **cannot** vary online to MVS the VTD addresses used by the CLINK connections as described in "How Clustered VTSS Configurations Work" on page 80.
- **In a Bi-Directional Cluster**, a VTV with replication enabled is allocated to either of the Peer VTSSs. If one of the two Peer VTSSs is either offline or quiesced, production workload can run on the remaining online VTSS. VTVs requiring replication, however, are allocated to the remaining VTSS only if no other Full-Function clusters are available and suitable. In this case, replicate VTVs are migrated immediately with keep and queued for replication when the other VTSS comes online.
- **In a Bi-Directional Cluster**, at dismount time, a VTV with replication enabled that resides on a Full-Function Cluster is queued for replication to the other Peer VTSS. If a VTV with replication enabled is dismounted from a VTD in a VTSS that is not part of a Full-Function Cluster, the VTV is queued for immediate migration. **Note that** the replication requirements of data is determined following a dismount, **not** a recall. Merely recalling a VTV will not cause a replicate – so demand recall, MVCdrain and reclaim will not cause a replicate. However, if the VTV is recalled and mounted on a VTD, at dismount time it will be replicated to the Secondary VTSS unless you specify REPLICAT(CHANGED) (the recommended option), which will cause the VTV to be replicated again only if the data is changed.
- **Both Peer VTSSs** can manage all space reclamations.
- If you are using ESCON or FICON interfaces:
  - On the each peer VTSS, the "sending" CLINK CIPs/FIPs are configured in **Nearlink Mode**, while the receiving CLINK CIPs/FIPs are configured in **Host Mode**.

    Therefore, you configure "sending" CLINKs on each Peer VTSS, as shown in the example in FIGURE 6-4, where VSMPR1 and VSMPR2 are Peer VTSSs.

```
.
.
CLUSTER NAME=CLUSTER1 VTSSs(VSMPR1,VSMPR2)
 CLINK VTSS=VSMPR1 CHANIF=0O:0
CLINK VTSS=VSMPR1 CHANIF=0O:1
 CLINK VTSS=VSMPR2 CHANIF=1O:0
CLINK VTSS=VSMPR2 CHANIF=1O:1
.
.
```

FIGURE 6-4 ESCON/FICON Bi-Directional Cluster and CLINK Definitions

- Each CLINK **must be attached to the same Storage Cluster** on each VTSS (Storage Cluster 0 to Storage Cluster 0 or Storage Cluster 1 to Storage Cluster 1). Failure to configure in this manner can produce Replicate, Channel, and Communication errors!

  As shown in the example in FIGURE 6-5, the sending (Nearlink mode) CLINK port on VSMPR1 is on Storage Cluster 1, and it connects to a receiving (Host Mode) CLINK port, **also on Storage Cluster 1** on VSMPR2. Similarly, a sending CLINK port on Storage Cluster 0 of VSMPR2 connects to a receiving CLINK port on Storage Cluster 0 of VSMPR1.

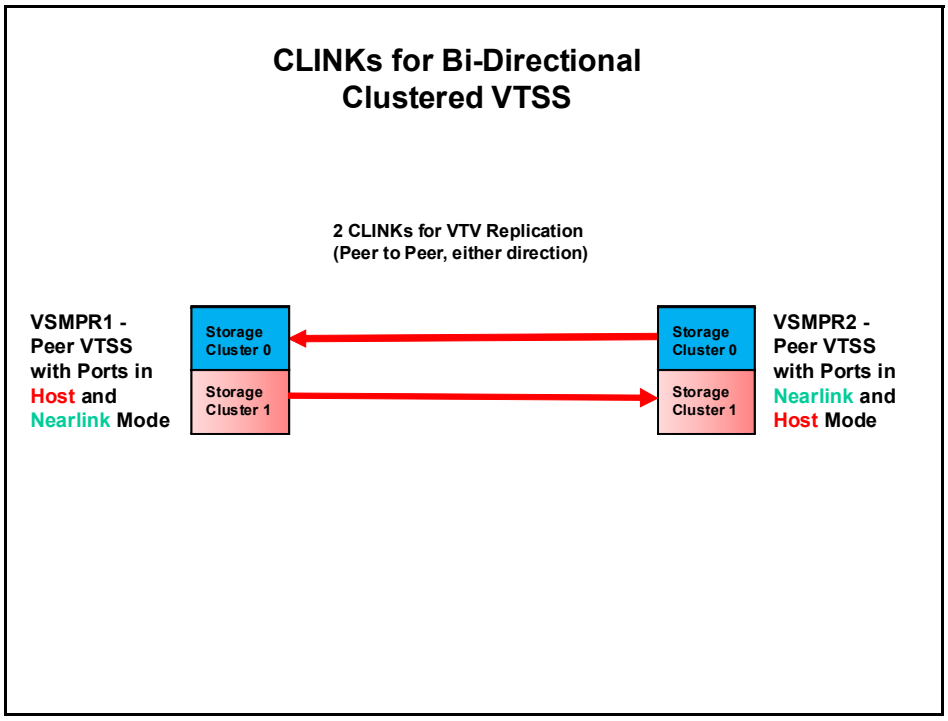

FIGURE 6-5 ESCON/FICON CLINKs for Bi-Directional Clustered VTSS

Extended Clustering

Extended Clustering (EC) allows three or more VTSSs to be connected by Clinks within a single Tapeplex (1 CDS) configuration. Clustering is a high-availability solution designed such that workload can continue without disruption on a VTSS outage. Clustering requires that all VTSS subsystems that are part of a cluster have access to all MVCs generated by any single VTSS subsystem in that cluster. If a VTSS within a cluster connects to a remote Tapeplex (CTR) then all VTSS systems in the cluster must connect to the same Tapeplex to retain the HA capability.

With Extended Clustering one VSM can be configured with Clinks connected to multiple VTSSs and the number of Clink connections are only limited by the number of physical connections available. D02.07.00.00 or greater micro code is required. All of the clustering and replication rules available are applied to EC. All Extended Cluster configurations are built based upon the two basic Uni-directional configurations shown in FIGURE 6-6.

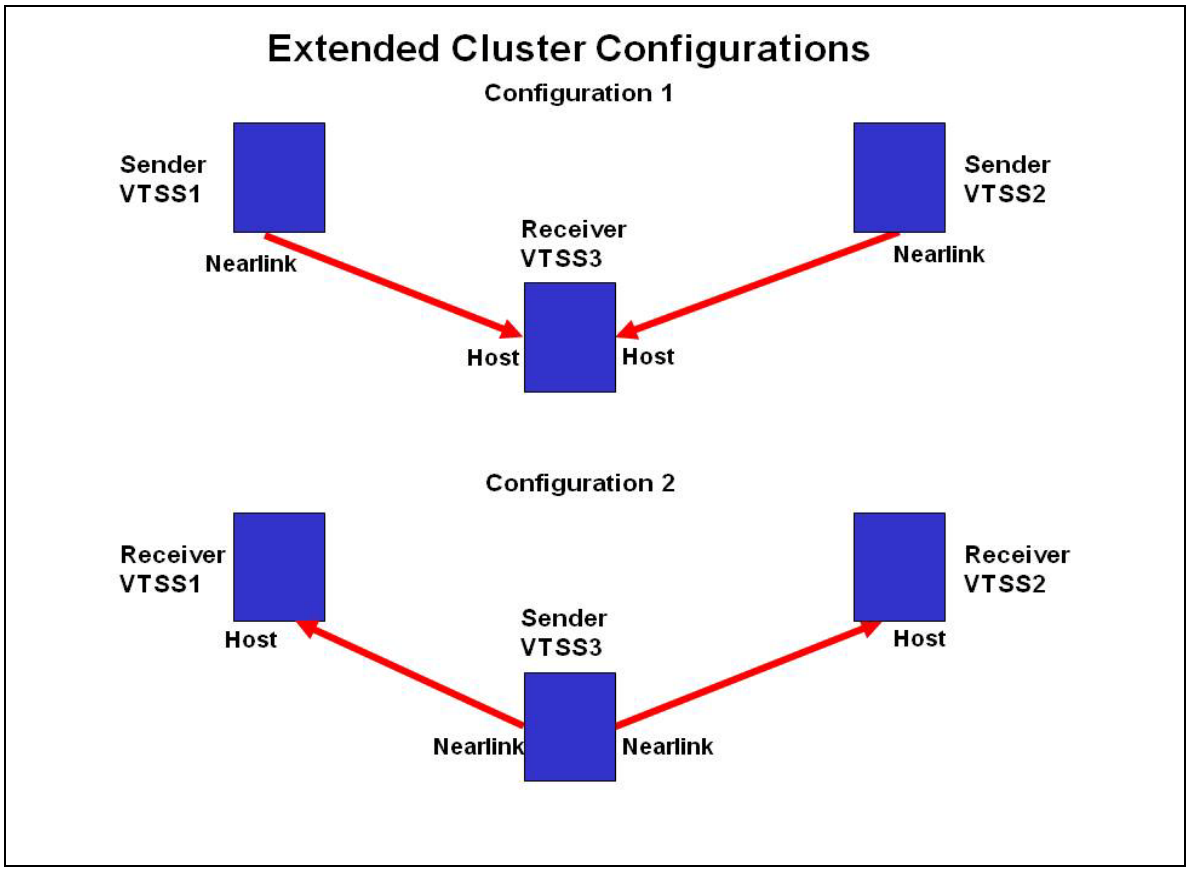

FIGURE 6-6 Basic Extended Cluster Configurations

Synchronous or Asynchronous Replication

You have a choice: you can either replicate synchronously or asynchronously, depending on your site's policies. **Please note** the following, however:

Caution – With synchronous replication the time required to replicate a virtual volume will delay the completion of any job creating data that has a synchronous replication policy.

Implementing Synchronous Replication

Caution – With synchronous replication the time required to replicate a virtual volume will delay the completion of any job creating data that has a synchronous replication policy.

1. **Ensure that your system has the Synchronous Replication requirements described in** TABLE 6-2 on page 79**.**

2. **With all HSC/VTCS systems down, use CONFIG GLOBAL to enable Synchronous Replication:**

   CONFIG GLOBAL SYNCHREP=YES

3. **Ensure that the CONFIG GLOBAL REPLICAT parameter is set as desired:**

   ALWAYS

   The replicate request is added to the VTCS replication queue every time the VTV is dismounted, regardless of whether the VTV was changed while it was mounted (the default).

   CHANGED

   The replicate request is added to the VTCS replication queue if the VTV:

   - Was changed while it was mounted **or**
   - Was only read while mounted but less than the expected number of MVC copies of the VTV exist.

4. **Specify Synchronous Replication on the desired MGMTClas statements:**

   MGMT (*name*) ..... REP(YES_SYNC)

Implementing Asynchronous Replication with Job Monitoring

You may elect to use Asynchronous Replication but may also want to know that the replication completed successfully. In this procedure, we'll use the DRMONitr utility to monitor to pause the associated MVS job until the replication completes successfully.

1. **Ensure that your system has the Synchronous Replication requirements described in** TABLE 6-2 on page 79**.**

2. **With all HSC/VTCS systems down, use CONFIG GLOBAL to enable Asynchronous Replication:**

   CONFIG GLOBAL SYNCHREP=NO

3. **Ensure that the CONFIG GLOBAL REPLICAT parameter is set as desired:**

   ALWAYS

   The replicate request is added to the VTCS replication queue every time the VTV is dismounted, regardless of whether the VTV was changed while it was mounted (the default).

   CHANGED

   The replicate request is added to the VTCS replication queue if the VTV:

   - Was changed while it was mounted **or**
   - Was only read while mounted but less than the expected number of MVC copies of the VTV exist.

4. **Specify Asynchronous Replication on the desired MGMTClas statements:**

   MGMT (*mgmtname*) ..... REP(YES)

5. **Create JCL to monitor the asynchronous replication.**

   For this, we use the DRMONitr utility to monitor the replication. DRMONitr causes the associated MVS job to pause until the replication completes successfully. For example:

```
//MONITOR EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//* If HSC IS NOT OR MAY NOT BE ACTIVE, INCLUDE THE
//* FOLLOWING:
//SLSCNTL DD DSN=primary.cds.name,DISP=SHR
//SLSCNTL2 DD DSN=secondary.cds.name,DISP=SHR
//SLSSTBY   DD DSN=standby.cds.name,DISP=SHR
//SLSPARMP DD DSN=hlq.PARMLIB(BKPCNTL),DISP=SHR
//SLSPARMS DD DSN=hlq.PARMLIB(BKPCNTL2),DISP=SHR
//SLSPARMB DD DSN=hlq.PARMLIB(BKPSTBY),DISP=SHR
//SYSIN         DD UNIT=SYSDA,SPACE=(TRK,1)
//* THE FOLLOWING IS USED BY THE SNAPSHOT UTILITY:
//SYSPRINT  DD  SYSOUT=*
//SLSPRINT   DD SYSOUT=*
//SLSIN        DD *
DRMON MGMT(mgmtname) REPL MAXAGE(24) TIMEOUT(120)
```

   In this example, the DRMON utility monitors the replicates for the specified Management Class. Additionally, monitor only VTVs that have been updated in the last 24 hours, and time out DRMON after 120 minutes.

Clustering with TCP/IP Connections

The VTSS native IP connection feature lets you use TCP/IP protocol to "cluster" (connect) two or more VTSSs for VTV replication. With Native IP clustering, each VTSS has Ethernet ports for connection to the TCP/IP network. Previously, you were limited to ESCON or FICON connections for replication. Using TCP/IP for CLINKs can provide improved replication performance over ESCON or FICON protocols and, if so desired, allows the existing ESCON or FICON ports to be used exclusively for RTD and host connections, where the following are supported:

- VSM5 to VSM5
- VSM5 to VSM 6
- VSM 6 to VSM 6

This section describes **only** the VTCS implementation for Native IP. Your StorageTek hardware support personnel or other QSPs are responsible for the VTSS side configuration.

The TCP/IP Environment

TCP/IP attached CLINKs perform the same function as FICON or ESCON channel attached CLINKs, but TCP/IP CLINK connect via an Ethernet port on the VTSS instead of from an ESCON or FICON port. The example in FIGURE 6-7 shows Peer VSM5s, each with 4 IFF3 cards with Ethernet ports. The Ethernet cables from the Ethernet ports on the IFF3 cards attach to Local Area Networks (LANs, one for each VTSS) and the LANs are connected via a Wide Area Network (WAN).

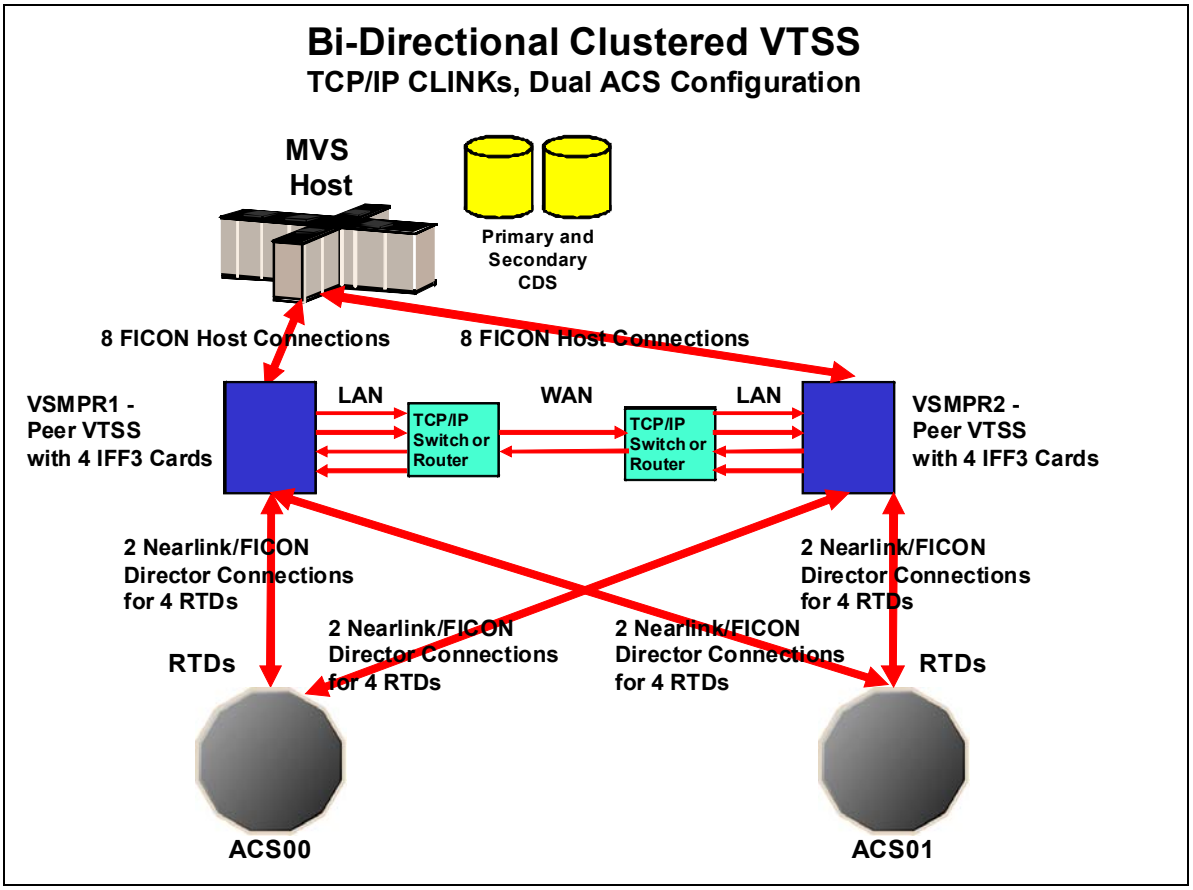

FIGURE 6-7 The TCP/IP Environment with Two VSM5s

Configuring VTCS for TCP/IP CLINKs

CONFIG CLINK Statement

The CONFIG CLINK statement provides two types of VTSS-to-VTSS connections via the following parameters:

CLINK CHANIF=*nn* or *nn:n*

defines a FICON or ESCON port for use as a CLINK.

CLINK IPIF=*ci:p*

defines an Ethernet port for use as a CLINK. Valid values for CONFIG RTD IPIF are *c*=0 or 1, *i*=A or I, *p*=o through 3 for VSM5s and VSM 6s. For VSM5s, this value must match the values specified on the VSM5 IFF Configuration Status Screen. For VSM 6s, this must be unique for each VTSS; and **does not** correspond to an actual value on the VSM 6 TCP/IP ports!

Note – CLINK statement must contain either the CHANIF or the IPIF parameter, but not both.

CHAPTER 7

Using the Concurrent Disaster Recovery Test Software

Customers that use or maintain a Disaster Recovery (DR) site as part of a business continuance plan may want to periodically validate their ability to continue normal production processing before an actual disaster occurs; other customers don't have a choice and must periodically demonstrate the readiness of their business continuance model to satisfy insurance requirements and/or their auditors.

Using the Concurrent Disaster Recovery Test (CDRT) facility, now an integrated feature of StorageTek ELS software, those businesses currently using StorageTek Streamline and/or Nearline (real hardware) tape libraries, VSM (virtual hardware), and associated software (HSC, VTCS) can validate their real and virtual tape business continuance capability without the need to purchase additional hardware and software.

CDRT supports a parallel test of production hosts and applications with simultaneous access to production data by both the production and the DR test systems.

Key CDRT concepts include the following:

- Using CDRT, a DR test can execute with real hardware, virtual hardware, or both.
- CDRT, HSC, and VTCS programmatically enforces certain functional restrictions during the CDS preparation and actual DR test in an attempt to ensure system integrity.
- CDRT logically separates a portion of existing production real and virtual hardware and tape volume pools for the period of the DR test. This allows testing of your DR configuration while concurrently running production work, ensures the integrity of the production data, and minimizes conflicts for tape volumes and hardware resources.

- CDRT creates a test copy of the production CDS. The production ELS subsystem and the DR test ELS subsystems therefore do not communicate with each other. Changes that occur in the DR test CDS are not reflected in the production CDS copy and vice versa. The DR test hosts exercise the logically separated hardware only. The production hosts continue to use all hardware with one exception: the DR hosts have exclusive use of any logically separated VTSSs during the DR test. Other resources, like RTDs, Multiple Virtual Cartridges (MVCs) and real scratch tapes, must be controlled by defining separate pools to each set of hosts.
- A DR test can be conducted using local resources only or with a combination of local and remote resources; configurations consisting of a remote site with real and virtual hardware only, or real and virtual hardware with a mainframe processor, are also supported.
- The DR test hardware may include any combination of ACSs, VTSSs, and VLEs.
- At the end of a DR test, the test copy of the CDS and all data created from the DR test are typically discarded and the logically separated hardware is redeployed back to the normal production environment.

Note –

- To satisfy the requirements of recovering from a true disaster, it is critical that jobs in a DR test jobstream **do not** update any volumes created in production, either through DISP=MOD or by overwriting these volumes. The use of such practices means that if a true disaster occurred, the state of these volumes would be unpredictable.
- For the DR test run, it is **strongly recommended** that production volumes that could be modified during the DR test are copied to new volumes at the beginning of the test, and that the **copied** volumes are updated by the DR test, not the original volumes. Also, JCL should be modified if possible so that the status of all tape volumes at the time of a disaster is known.

Metadata Considerations

Fundamental to the execution of a successful DR test using CDRT is a consistent copy of the state of all tape volumes managed by ELS software and the real and virtual hardware. The consistency in the state of tape volumes between the production hosts and the DR hosts at the start of the DR test is what allows the parallel processing of customer applications. Since the CDS reflects the state of all tape volumes and resources in the real and virtual hardware, CDRT partially meets this consistency requirement for you when it makes its test copy of the CDS.

In a tape volume environment, however, quite often some of this tape volume state data (metadata) is retained and managed outside of the ELS subsystem and the real and virtual hardware. Typically, tape volume metadata (i.e. VOLSER, DSN, Expiration date, scratch status, real or virtual designation, etc.) is stored in one or more Tape Management Catalogs (TMCs), one or more z/OS catalogs and the CDS.

You must co-ordinate the creation of copies of metadata retained and managed outside of ELS (and the real and virtual hardware) with the creation of the test copy of the CDS by CDRT.

Where Does CDRT Get Its VTV Data?

CDRT gets its VTV data from one or more of the following DR site resources:

- MVCs
- VLEs
- VTSSs

Because the production CDS is the source of information about VTVs available to the DR test, it is important to ensure that the scratch synchronization cycle allows that volumes used in the DR test are not put into scratch status before the start of the test. StorageTek also recommends that you vary your test VTSS offline before doing your DRTEST CREATE.

Note that running scratch during the test **does not** affect the contents of the DR test VTSS, nor of the MVCs, assuming that these are set to READONLY using the ACTMVCGN utility.

The DR site CDS copy provides the location of the VTVs at the time of the copy, and the location is usually on MVCs or VLEs. Sometimes, however, you may choose to use VTV copies in a VTSS that exists at the test site. In general, you can do this in the following situations:

- You define your DR VTSS with the SHARE keyword, which prevents updates to the contents of any production VTV
- You have two clustered VTSSs with one at the production site and one at the DR site **and** your DR test **does not modify** the contents of any production VTV
- You have two clustered VTSSs with one at the production site and one at the DR site **and** you are willing to spend time after your DR test to identify and manually migrate any VTV that was updated by the DR test.

Note – You can use the DRMONitr utility to ensure critical DR data reaches its designated recovery location before running a DR test.

What happens to data created by the DR test when the test is over?

Data that is created by the DR test is reflected **only** in the DR test CDS (which is discarded after the test) and on the DR test MVCs, which are in a segregated range of volumes from the production MVCs. In addition, VTVs created by the DR test remain on the DR test VTSS after the test, unless you do one the following:

1. Use the SLUADMIN SCRATCH utility in the DR test environment to scratch (with MGMTCLAS DELSCR(YES)) all VTVs in the DR test subpool range. This option requires the use of the POOLPARM/VOLPARM feature.
2. Ensure that all production VTVs that have been modified by the DR test are migrated to DR test MVCs using the MIGRATE command with DELETE(YES) from the DR test system. If you fail to do this, the production system to picks up data that has been modified by the DR test.

3. Have your StorageTek CSE or other QSP "clean" the DR test VTSS following the test to remove all data from the VTSS.

Note – If you use either Option 2 or Option 3, the contents of the VTSS **will not match** the production system, because VTVs are missing from the DR test CDS when it is returned to production. Although this condition is handled by the software, your production performance may be degraded.

Managing CDRT Resources

The following sections tell how to manage CDRT resources.

Volume Resources

The first step in managing CDRT resources is to define the volumes in your system using the POOLPARM/VOLPARM utility. This feature simplifies overall management of your tape pools and volumes, and provides a method of segregating volumes that are written to in a CDRT configuration.Using POOLPARM/VOLPARM is **required** when DR test VTSSs are shared, and is **strongly recommende**d in for other scenarios.

Note – The SLUADMIN SET VOLPARM utility must be run using the production CDS and must define both production and DR test pools. The SET VOLPARM utility **is not** valid for a DR test CDS.

Scratch Subpools

Scratch subpools are applicable to all DR test scenarios. The following syntax shows using the POOLPARM/VOLPARM definitions to define production and DR test scratch subpools. By using the same names for subpools in production and DR test (with different volume ranges), you do not have to change your production policies when running the DR test as shown in CODE EXAMPLE 7-1.

CODE EXAMPLE 7-1 POOLPARM/VOLPARM Scratch Subpools

```
*  SCRATCH POOLS
POOLPARM NAME(SCRP1) TYPE(SCRATCH)
VOLPARM VOLSER(T11000-T11999) MEDIA(T10000T1) RECTECH(T1AE)
POOLPARM NAME(SCRP1) TYPE(SCRATCH) DRTEST
VOLPARM VOLSER(T12000-T12999) MEDIA(T10000T1) RECTECH(T1AE)
POOLPARM NAME(SCRVTV1) TYPE(SCRATCH)
VOLPARM VOLSER(V1000-V1999) MEDIA(VIRTUAL)
POOLPARM NAME(SCRVTV1) TYPE(SCRATCH) DRTEST
VOLPARM VOLSER(V2000-V2999) MEDIA(VIRTUAL)
```

When you define scratch subpools using the POOLPARM/VOLPARM utility, you can use the HSC SLUADMIN SCRATCH utility to scratch DR test output volumes within the DR test environment. In combination with a Management Class that specifies DELSCR(YES), running the scratch utility removes VTVs created by the DR test from the VTSS.

MVC Resources

MVC resources are used for all DR test scenarios except for real tape only and tapeless VSM. CODE EXAMPLE 7-2 shows using POOLPARM/VOLPARM definitions to define production and DR test MVC pools.

CODE EXAMPLE 7-2 POOLPARM/VOLPARM MVC Pools

```
*  MVC POOLS
POOLPARM NAME(MVCP1) TYPE(MVC) MVCFREE(40) MAXMVC(4) THRESH(60) +
START(70)
VOLPARM VOLSER(T14000-T14999) MEDIA(T10000T1) RECTECH(T1AE)
POOLPARM NAME(MVCP1) TYPE(MVC) MVCFREE(40) MAXMVC(4) THRESH(60) +
START(70) DRTEST
VOLPARM VOLSER(T13000-T13999) MEDIA(T10000T1) RECTECH(T1AE)
```

Preserve the contents of production MVCs for use in a DR test scenario using the following:

- Use the ACTMVCGN utility to set MVCs that will be used as input to the DR test to READONLY as shown in CODE EXAMPLE 7-3.

CODE EXAMPLE 7-3 ACTMVCGN Utility Before the DR test

```
//ACTMVCG1 EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//* NOTE: MVCMAINT READONLY(ON) STATEMENTS
//SLUSMVON DD DSN=hlq.SLUSMVON,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,1)
//* NOTE: MVCMAINT READONLY(OFF) STATEMENTS
//SLUSMVOF DD DSN=hlq.SLUSMVOF,DISP=(NEW,CATLG,DELETE),
SPACE=(CYL,1)
//* NOTE: THE FOLLOWING STEP SELECTS ALL "ACTIVE" MVCS
//* IN ACS 01.
//SLSIN DD *
ACTMVCGN ACS(01)
/*
//ACTMVCG2 EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//* NOTE: EXEC MVCMAINT TO SET READONLY(ON)
//SLSIN DD DSN=hlq.SLUSMVON,DISP=SHR
```

- Ensure that reclaim is not run on all production hosts via VTCS CONFIG:

CONFIG HOST NAME(*host*) NORECLAM.

VTSS Resources

Non-shared VTSS resources

In an environment where VTVs are migrated to MVCs or to VLE, VTSS resources are segregated during a DR test. The DR test system has access only to VTSSs that are defined via the DRTEST PRIMEPRD/CREATE utilities and these VTSSs must remain offline to production. The DRTEST START command is rejected if a DR VTSS is online in the production environment.

In some cases, there may be two VTSSs with the same name in both the production and test sites. In this case, the SPARE parameter of the DRTEST PRIMEPRD/CREATE utility specifies that the DR test VTSS has the same name as a production VTSS but is not physically the same device, allowing the production device to remain online during the test. **Do not** use the SPARE parameter for any other scenarios!

Clustered VTSS in CDRT

"Scenario 4: Clustered VTSSs with Production and DR Test Sites" on page 126 uses clustered VTSSs to deliver data to the VTSS at the DR site. In this scenario, the cluster **does not** function during the DR test because the DR test VTSS is offline to production.

If production VTVs are modified in any way during the DR test, you **must** ensure that the modified VTVs are removed from the VTSS **before** varying the test VTSS back online to production. Otherwise, the modified VTVs can be picked up by the production site. To prevent this situation, StorageTek **strongly recommends** that the DR test is designed so that production VTVs are **not** altered by the test.

In a Clustered VTSS scenario there may be no VTDs in the test VTSS accessible to the production host. To allow ECAM communication from production to the DR test VTSS, specify the following in VTCS CONFIG:

VTD LOW=*xxxx* HIGH=*xxxx* NOVERIFY

To prepare your VTSS cluster for a DR test:

1. **Vary the DR test VTSS to quiesced state. For example:**

   VARY VTSS1 QUIESCED

   The objective here is to (gracefully) shut down replication to `VTSS1` so we can use it exclusively for the DR test.

2. **Monitor replication until it completes.**

   ...with `Display REPLicat`. Here, replication is still active:

   VTSS HOST QDEPTH

   VTSS0 PRODUCTION 1

   You know replication is complete when you see this:

   VTSS HOST QDEPTH

   VTSS0 PRODUCTION 0

3. **Cross check that replication is complete by checking the CLINK status...**

   ...with `Display CLINK`. Here, the CLINK is still active:

   VTSS CLINK STATUS USAGE HOST

   VTSS0 7 ONLINE REPLICATING PRODUCTION

   VTSS0 8 ONLINE REPLICATING PRODUCTION

   You know the CLINK is no longer active when you see this:

   VTSS CLINK STATUS USAGE HOST

   VTSS0 7 ONLINE FREE

   VTSS0 7 ONLINE FREE

4. **Vary the DR Test VTSS ofline:**

   VARY VTSS1 OFFLINE

Managing VTSS Contents Before And After the DR Test

Before you start a DR test, you **must**:

1. Plan your test so that test output does not accidentally get into production.

2. Prepare your test so that your DR test CDS matches the DR VTSS contents.

Additionally, StorageTek **strongly recommends** that all test site VTSSs are cleaned as soon as possible after the test. This practice ensures that the test site VTSSs are empty at the start of the next test and that no DR test data remains on VTSSs that are returned to production.

You can clean test VTSSs by doing either of the following:

1. Do not allow the DR test to perform any updates (overwrite or append) to the production VTVs, and use POOLPARM/VOLPARM to define DR test scratch subpools. With this method, you can run the scratch utility to scratch all VTVs in the DR test range and they are automatically deleted from the VTSS (if the Management Class specifies DELSCR(YES)).

2. If your DR test can update production VTVs, then you **must** ensure that any VTSS used for DR test output is emptied prior to returning the VTSS to production. This can be done by migrating to zero in the DR test environment, or by having the StorageTek CSE or other QSP "clean" the VTSS.

When you run the DRTEST CREATE utility to create the DR test CDS, metadata about the VTSS contents is propagated to the test environment. VTVs on the DR test VTSS are available to the DR test environment.

If the DR test VTSS is defined as a spare, then the contents of the physical DR VTSS will **not** match the CDS metadata, which references the production instance of the spare VTSS. To remove the discrepancy, before creating the DR test CDS, migrate the production instance of the spare VTSS to 0 in the production environment. You can run VTCS VTVRPT OPTION(UNAVAIL) to ensure that all VTVs are migrated and are available to other VTSSs. If this step is not performed, DR test attempts to access VTVs on the spare VTSS will issue SLS6680E messages for VTV mounts.

Shared VTSS Resources

If the tape environment does not include either real tape or VLE MVCs, You can run CDRT in a shared VTSS environment. The DRTEST PRIMEPRD/CREATE SHARE parameter specifies that a VTSS is shared between production and DR test during the test. This environment enforces the following restrictions:

1. The DRTEST CREATE cannot also define any DRACS or STORMNGR resources. That is, no data can be migrated from the shared VTSS to external media.
2. The production system must define volume resources using the POOLPARM/VOLPARM facility.
3. Mounts of a non-DRTEST subpool VTV in the DR test environment forces the VTV to be readonly. That is, the DR test is not allowed to overwrite or append to any production VTV.

ACS Resources

ACS resources are used for DR test scenarios with real tape only or with virtual tape without VLE in a non-tapeless VSM environment. ACS resources for CDRT are specified in the DRTEST PRIMEPRD/CREATE DRACS parameter.

DR Test ACS Restrictions

You must issue the HSC CAPPREF command to set the CAPs to manual prior to running the DRTEST CREATE utility. The software insures that they remain in that state as long as the test is in effect. After you enter the DRTEST START command, restrictions are enforced on the DR test ACS in both the production and in the DR test environment to ensure consistency between the hardware and the respective CDSs to permit both sites to access the data. The DRTEST START command is rejected if a CAP in a DR test ACS is in automatic mode in the production environment.

The following restrictions are automatically enforced by the software.

On the production hosts during an active DR test for the DR ACS

- CAPs must remain in manual mode.
- FLOAT(OFF) and EJCTAUTO(OFF) are automatically enforced, regardless of the settings in the MNTD command
- You cannot run eject, move, audit, and scratch utilities for the DR test ACS.

On the DR test hosts during a DR test

- Non-DR test ACSs cannot be varied online.
- CAPs must remain in manual mode.
- FLOAT(OFF) and EJCTAUTO(OFF) automatically enforced and the other values are not valid on the MNTD command.
- You cannot run eject, move, audit, and scratch utilities for the DR test ACS.
- Scratch updates are allowed only for volumes defined using POOLPARM/VOLPARM as DR test scratch subpool volumes.

VLE Resources

You define VLE resources via STORMNGR parameter of the in the DRTEST PRIMEPRD and CREATE commands. Typically, the DR test resources are either ACSs or VLEs, although there is no restriction on using both.

Like physical MVCs, VLE VMVCs are defined using the POOLPARM/VOLPARM facility, with a range of VMVCs reserved for the DR test. The DR test also has read access to the production VMVCs as shown in CODE EXAMPLE 7-4.

CODE EXAMPLE 7-4 ACTMVCGN Utility Before the DR test (VLE)

```
//ACTMVCG1 EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//* NOTE: MVCMAINT READONLY(ON) STATEMENTS
//SLUSMVON DD DSN=hlq.SLUSMVON,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,1)
//* NOTE: MVCMAINT READONLY(OFF) STATEMENTS
//SLUSMVOF DD DSN=hlq.SLUSMVOF,DISP=(NEW,CATLG,DELETE),
SPACE=(CYL,1)
//* NOTE: THE FOLLOWING STEP SELECTS ALL "ACTIVE" MVCS
//* IN VLE1 AND MVCPOOL MVCP1.
//SLSIN DD *
ACTMVCGN STORMNGR=VLE1,MVCP=MVCP1
/*
//ACTMVCG2 EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//* NOTE: EXEC MVCMAINT TO SET READONLY(ON)
//SLSIN DD DSN=hlq.SLUSMVON,DISP=SHR
```

VTCS Policies

To minimize the differences between your production and DR test environments, CDRT allows you to define policies for managing VTVs that have the same name in both production and DR test environments, but different policy definitions.

Note that the production and test sites can share a VTSS via the DRTEST DRVTSS SHARE parameter. Also note that the specifying a dummy DR ACS is no longer required for a shared VTSS or a DR test using VLE. Specifying a shared VTSS has the following restrictions:

- The shared DR test VTSS should not have active RTD connections from either the production or DR test site.
- The CDS must contain VOLPARM definitions to define DR test scratch subpools.
- Production VTVs (those not in a DR test subpool) are set to read-only when they are mounted, and cannot be modified.

Defining MGMTCLAS/STORCLAS for Non-Shared VTSSs

When you define management classes for a DR test system, you normally define only a single MVC copy of the output VTVs. To allow VTSS cleanup after the test, it is recommended that DELSCR(YES) be specified as shown in the example in CODE EXAMPLE 7-5

CODE EXAMPLE 7-5 STORCLAS/MGMTCLAS for Non-Shared VTSSs

```
STOR NAME(LOCAL) ACS(01) MVCPOOL(MVCP1)
MGMT NAME(CRITICAL) MIGPOL(LOCAL) IMMWAIT(0) DELSCR(YES)
```

Defining MGMTCLAS for Shared VTSSs

When you use a shared VTSS for the DR test, no migrated output copies are allowed. You must specify DELSCR(YES) to allow cleanup following the test as shown in the example in CODE EXAMPLE 7-6.

CODE EXAMPLE 7-6 STORCLAS/MGMTCLAS for Shared VTSSs

```
MGMT NAME(CRITICAL) DELSCR(YES)
```

Optimizing Access to Test and Production Resources

During a DR test, it is recommended that you enforce procedures to optimize access to resources in both test and production environments. Specifically:

- Before starting the DR test, define production management classes that specify immediate migration to both production and DR test ACSs so that VTVs will be available on MVCs accessible to the DR test system as well as the production system.
- Define DR test management classes that specify a single migration copy, since only one ACS is normally available to the DR test site.
- Use the POOLPARM/VOLPARM facility to segregate both scratch subpools and MVC pools between production and DR test.
- If possible, ensure that your DR test processing does not update any pre-existing VTVs (DISP=MOD or overwrite with DISP=OLD).
- Minimize contention between production jobs migrating to the DR test ACS and DR test jobs accessing VTVs on MVCs in the DR test ACS by running the ACTMVCGN utility to mark active MVCs as read-only in the production environment.
- Disable MVC space reclamation (via CONFIG HOST NORECLAM) on production MVCs during DR test, to preserve the MVC contents of volumes being used by the DR test system.

Running a DR Test

Note – For more information about the commands and utilities used in this procedure, see *ELS Command, Control Statement, and Utility Reference*.

▼ To run a DR test:

1. **Define volume pools in the production CDS using the** SET VOLPARM **command and** POOLPARM/VOLPARM **statements in the** SLSPARM DD**.**

   For more information, see "Volume Resources" on page 101.

2. **Ensure that your test resources are set correctly.**

   For more information, see:

   - "ACS Resources" on page 105
   - "VTSS Resources" on page 103
   - "VLE Resources" on page 106

3. **Create** MGMTCLAS/STORCLAS **statements for the DRTEST environment.**

   For more information, see

   - "Defining MGMTCLAS/STORCLAS for Non-Shared VTSSs" on page 107
   - "Defining MGMTCLAS for Shared VTSSs" on page 107

4. **Copy the MVS Catalog for the DR test site if required.**

5. **Optionally, copy the TMS database (if a TMS is used) for the DR test site.**

6. **At the Production Site, run the** DRTEST **utility (with the** PRIMEprd **keyword) to prepare the production CDS for the DR test.**

   For example, see CODE EXAMPLE 7-9 on page 116.

   You only need to run PRIMEprd **once** in your environment, no matter how many DRTEST iterations you run, **unless** your configuration changes. If your DR test configuration changes in any way, then you need to rerun PRIMEprd. Note also that you do not have to run the DRTEST RESET utility after the DR test is complete. Although flags remain set in the production CDS, they have no effect on processing as long as the DR test is not active.

7. **On the production system, use the HSC** CAPPREF **command to set all CAPs in the DR test ACS to manual mode.**

8. **At the DR Test Site, run the** DRTEST **utility (with the** CREATE **keyword) against the mirrored or backup copy of the production CDS to create the new DR Test CDS for the DR test.**

   Each scenario provides a DRTEST CREATE example.

   The CDSs must be allocated via the DD statements on the utility. **Note that** when NOUPD is used, only the SLSCNTL DD statement is required, and it can be either the actual primary CDS, a backup, or a mirrored copy.

9. **Start the DR test at the production site, pointing to the DRTEST MGMTCLAS/STORCLAS definitions you created in Step 3.**

   For example:

   CODE EXAMPLE 7-7 Starting the DR Test

```
/PRIME EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB DD DSN=hlq.SEALINK,DISP=SHR
//SLSIN DD *
DRTEST START
```

   Note – You can also start the test by entering a DRTEST START command from the console.

10. **Start the SMC system on the DRTEST client host(s).**
11. **Start the SMC/HSC/VTCS system(s) (pointing to the CDS you created in Step 8) on the DR test systems.**
12. **Verify the VTDs for the test VTSS(s) and paths are online.**
13. **Vary online the DR VTSSs to the DR system.**
14. **If applicable, vary online the DR RTDs to the DR system.**
15. **Run the tests at the DR test site.**

    During the DR test, the following conditions are programmatically enforced:

    - The production site ACS(s) are disconnected from the DR test host(s).
    - The production site VTSSs are offline to the DR test host(s).
    - No floating dismounts, ejects, moves, scratch updates, audits, or scratch redistributions can occur at the DR test site.
    - No floating dismounts, enters/ejects, moves, audits, or scratch redistributions on the DR test ACS can occur at the production site.
    - All CAPs in the DR test ACS are in manual mode.

    Note – You can enter volumes into the DR Test ACS, but after the test is complete, you must either eject the volumes or audit the cells to synchronize the production CDS with the actual library volumes.

Cleaning Up After a DR Test

As described on page 98, it is critical that jobs in a DR test jobstream **do not** update any volumes created in production.

Note – For information about the DRTEST command and DRTEST utility, see *ELS Command, Control Statement, and Utility Reference*. For information about CDRT messages, see *ELS Messages and Codes*.

▼ Cleaning Up After a DR Test

1. **Run the SLUADMIN SCRAtch utility to scratch all possible VTVs in your DRTEST subpool(s).**

   Because you set DELSCR(YES) in your Management Class, the VTVs will be automatically deleted from the buffer when you scratch them at the conclusion of the test.

   Warning. If you do not use SET VOLPARM and you do not set up separate scratch pools, you are risking data loss!

   ---

2. **If your DR test has modified or may have modified any production VTVs, you must also do the following to ensure that no DR test data remains on the production VTSS:**
   - Run a VTV report on the DRTEST CDS, and examine the output to determine if any VTVs in the production VTV ranges were modified during the test.
   - If any VTVs were modified, you must do one of the following:
     - Demand migrate the modified VTVs based on the VTV report.
     - Migrate the DR test VTSS to 0.
     - Have the CSE "clean" the test VTSS.
3. **Stop HSC/VTCS and SMC on the `DR TEST` MVS system.**
4. **If the ACTMVCGN utility was run prior to the test to put MVCs into a READONLY state, then run SLUADMIN using the output of the SLUSMVOF DD statement as input to reset the READONLY state.**

▼ To resume normal operations:

1. **Stop the DR test on the** `PRODUCTION` **MVS system.**

   For example:

   CODE EXAMPLE 7-8 Resume Normal Operations

   ```
   /STOP EXEC PGM=SLUADMIN,PARM='MIXED'
   //STEPLIB DD DSN=hlq.SEALINK,DISP=SHR
   //SLSPRINT DD SYSOUT=*
   //SLSIN DD *
   DRTEST STOP
   ```

   Note also that you do not have to run the DRTEST RESET utility after the DR test is complete. Although flags remain set in the production CDS, they have no effect on processing as long as the DR test is not active

2. **If desired, place CAPs in automatic mode.**

Operational Scenarios

This section tells how to use the DR Test software to set up the environment for, start, and stop DR testing. This section consists of the following information:

For information about the DRTEST command and DRTEST utility, see *ELS Command, Control Statement, and Utility Reference*. For information about CDRT messages, see *ELS Messages and Codes*.

Note – For **all** scenarios, after the test, run the procedure in "Cleaning Up After a DR Test" on page 111.

Scenario 1: Production and Test Sites, ACS at Each Site, Spare VTSS at Test Site

In Scenario 1, there is a single ACS at both the production and test sites, and “spare” VTSS(s) at the test site used solely for testing (no requirements to migrate or restore “spare” VTSS contents). In normal operations, the production site writes and accesses VTVs on VTSSs at the production site, and output VTVs are always migrated immediately and duplexed to separate MVCs, one in each ACS. FIGURE 7-1 on page 115 shows the system for Scenario 1 before running the `DRTEST` utility.

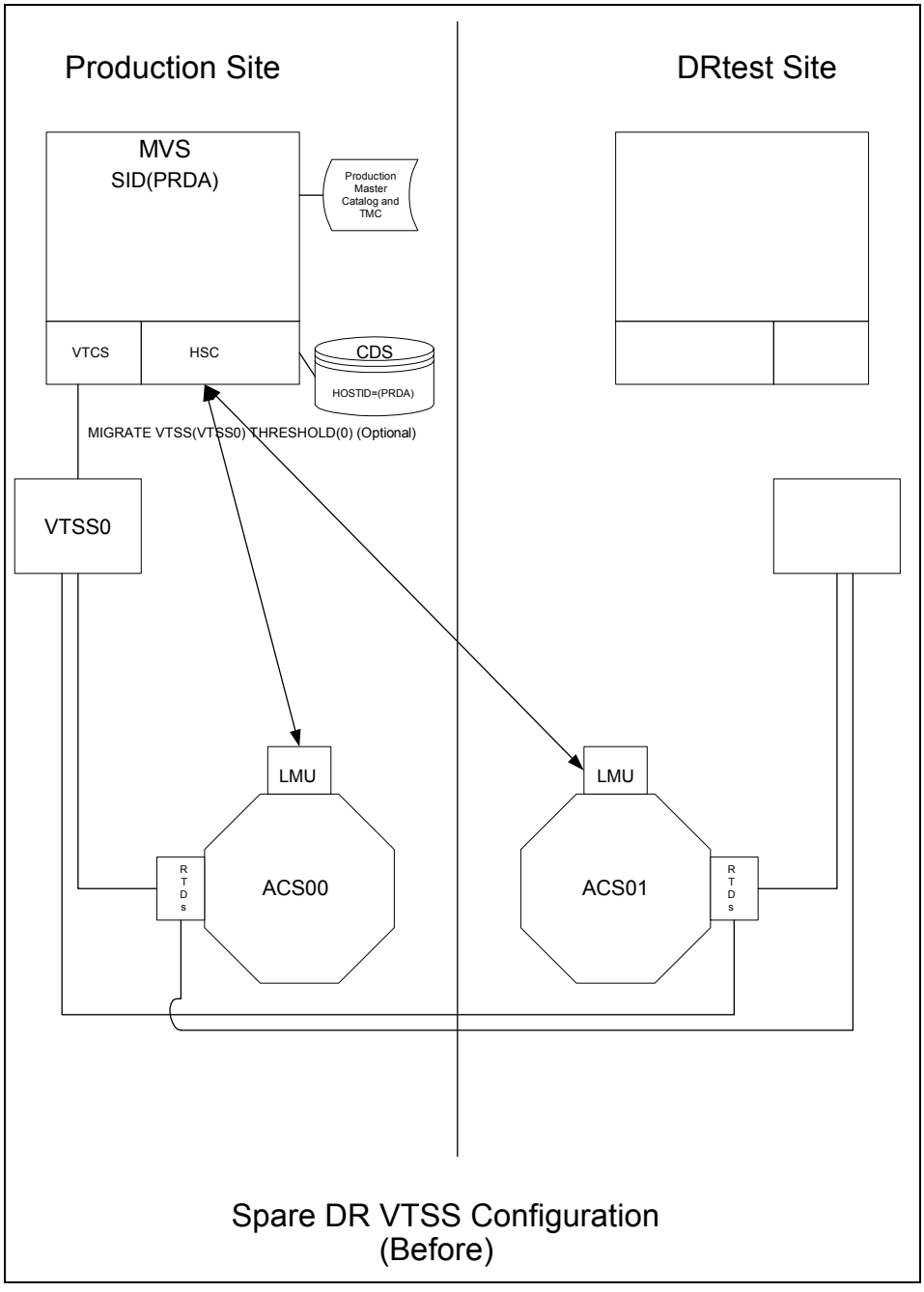

FIGURE 7-1 Spare VTSS Configuration - Before Running the DRTEST Utility

Example JCL for Scenario 1

CODE EXAMPLE 7-9 Scenario 1, PRIMEPRD Step

```
//DRTPRIME EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSIN DD *
DRTEST PRIMEPRD +
DRACS(01) DRVTSS(VTSS0) SPARE HOST(MVS1,MVS2)
```

CODE EXAMPLE 7-10 Scenario 1, CREATE Step

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
DRACS(01) DRVTSS(VTSS0) SPARE HOST(MVS1,MVS2)
```

FIGURE 7-2 shows the system for Scenario 1 (spare VTSS at test site) after running the DRTEST utility.

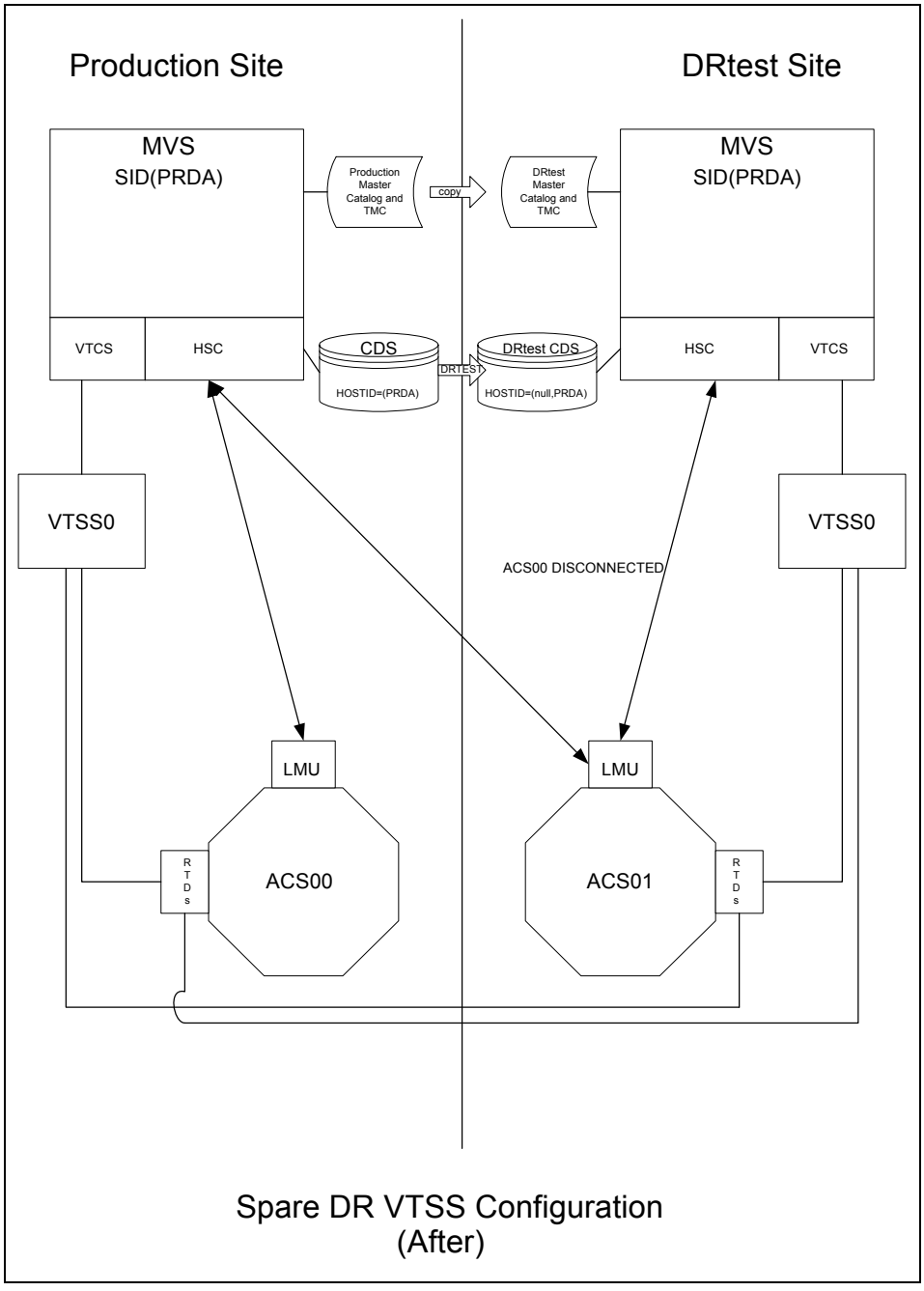

FIGURE 7-2 Spare VTSS Configuration - After Running the DRTEST Utility

Scenario 2: Production and Test Sites, ACS at Each Site, VTSS Takeover at Test Site

In Scenario 2, there is a single ACS at both the production and test sites, but no “spare” VTSS(s) at the test site used for testing. In normal operations, the production site writes and accesses VTVs on VTSSs at both sites, and output VTVs are always migrated immediately and duplexed to separate MVCs, one in each ACS. In this configuration, you must demand migrate to zero one or more VTSSs at the test site and vary these VTSSs offline to the production system, so testing can take over the required VTSS resources. In addition, one or more LPARs at the test site function as displaced production systems, running in parallel with the actual production systems. Both ACSs are online to the production system.

FIGURE 7-3 on page 119 shows the system for Scenario 2 (VTSS takeover at test site) before running the `DRTEST` utility.

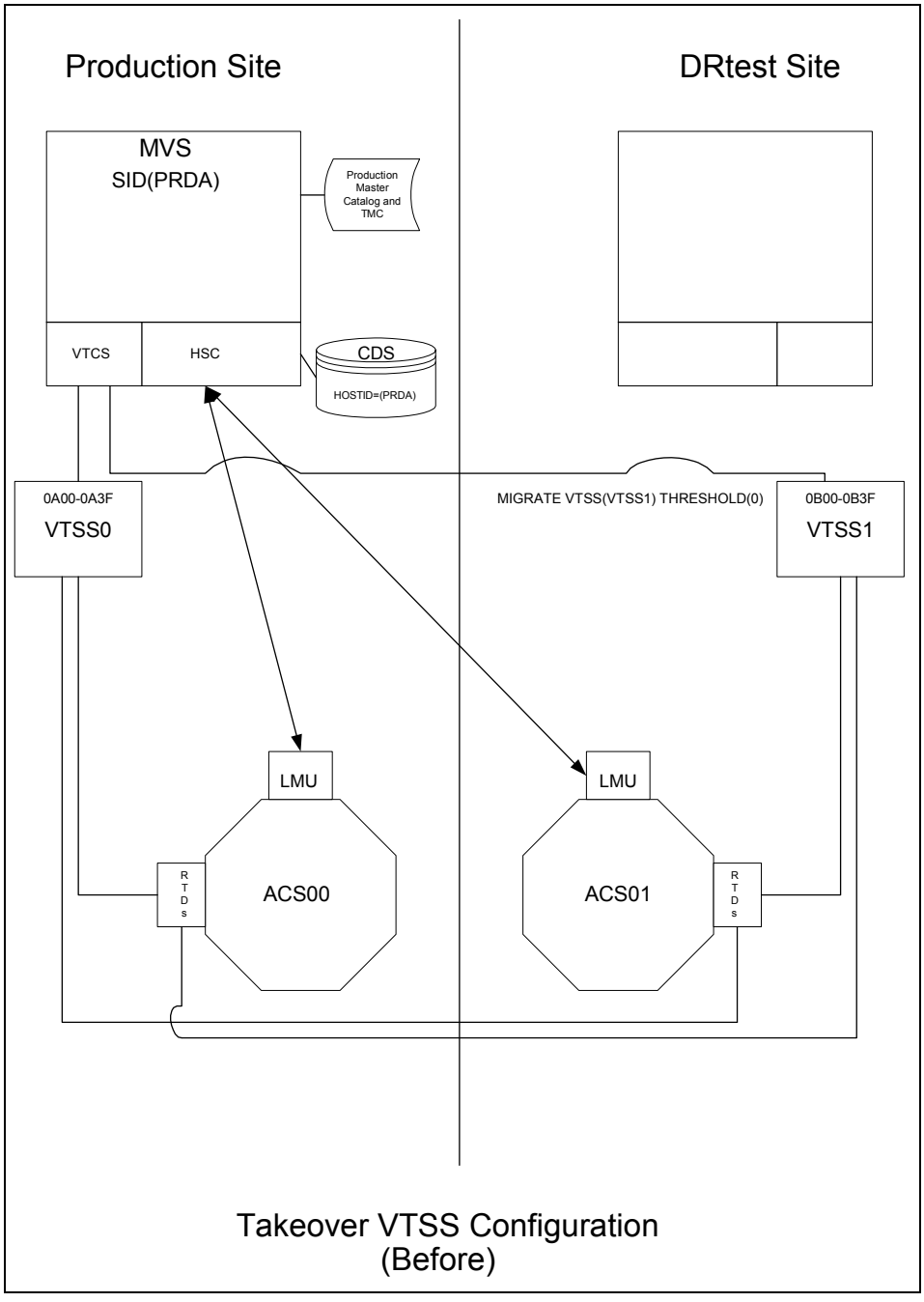

FIGURE 7-3 VTSS Takeover Configuration - Before Running the DRTEST Utility

Example JCL for Scenario 2

CODE EXAMPLE 7-11 Scenario 2, CREATE Step Only, PRIMEPRD Run Previously

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
DRACS(01) DRVTSS(VTSS1) HOST(MVS1,MVS2)
```

FIGURE 7-4 shows the system for Scenario 2 (VTSS takeover at test site) after running the DRTEST utility.

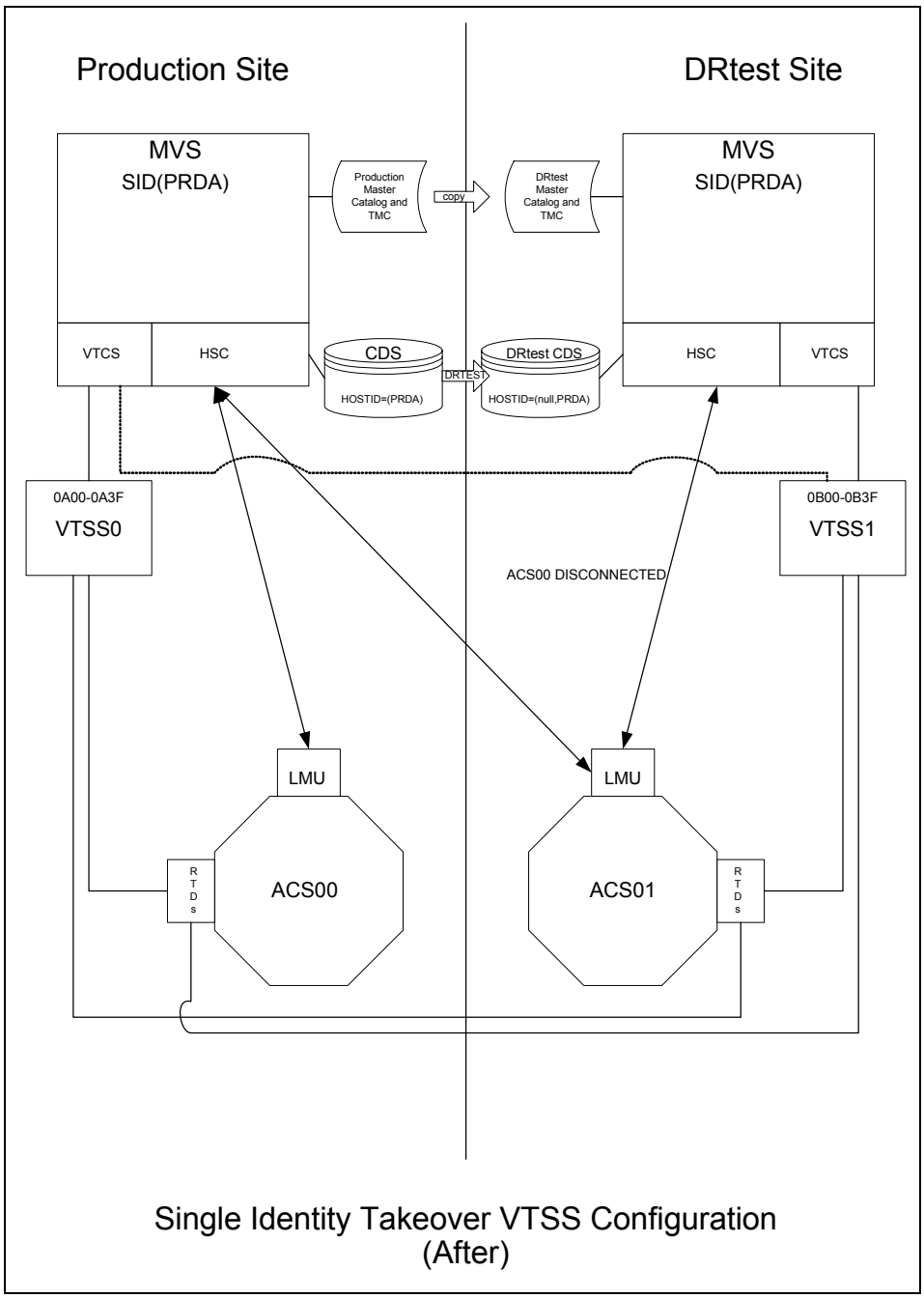

FIGURE 7-4 VTSS Takeover Configuration - After Running the DRTEST Utility

Scenario 3: Production and Test Sites, ACS at Each Site, No VTSSs

In Scenario 3, there is a single ACS at both the production and test sites, but no VTSS(s) at the test site used for testing. In normal operations, the production site writes and accesses tape data sets at both sites. FIGURE 7-5 on page 123 shows the system for Scenario 3 (real-only configuration) before running the `DRTEST` utility.

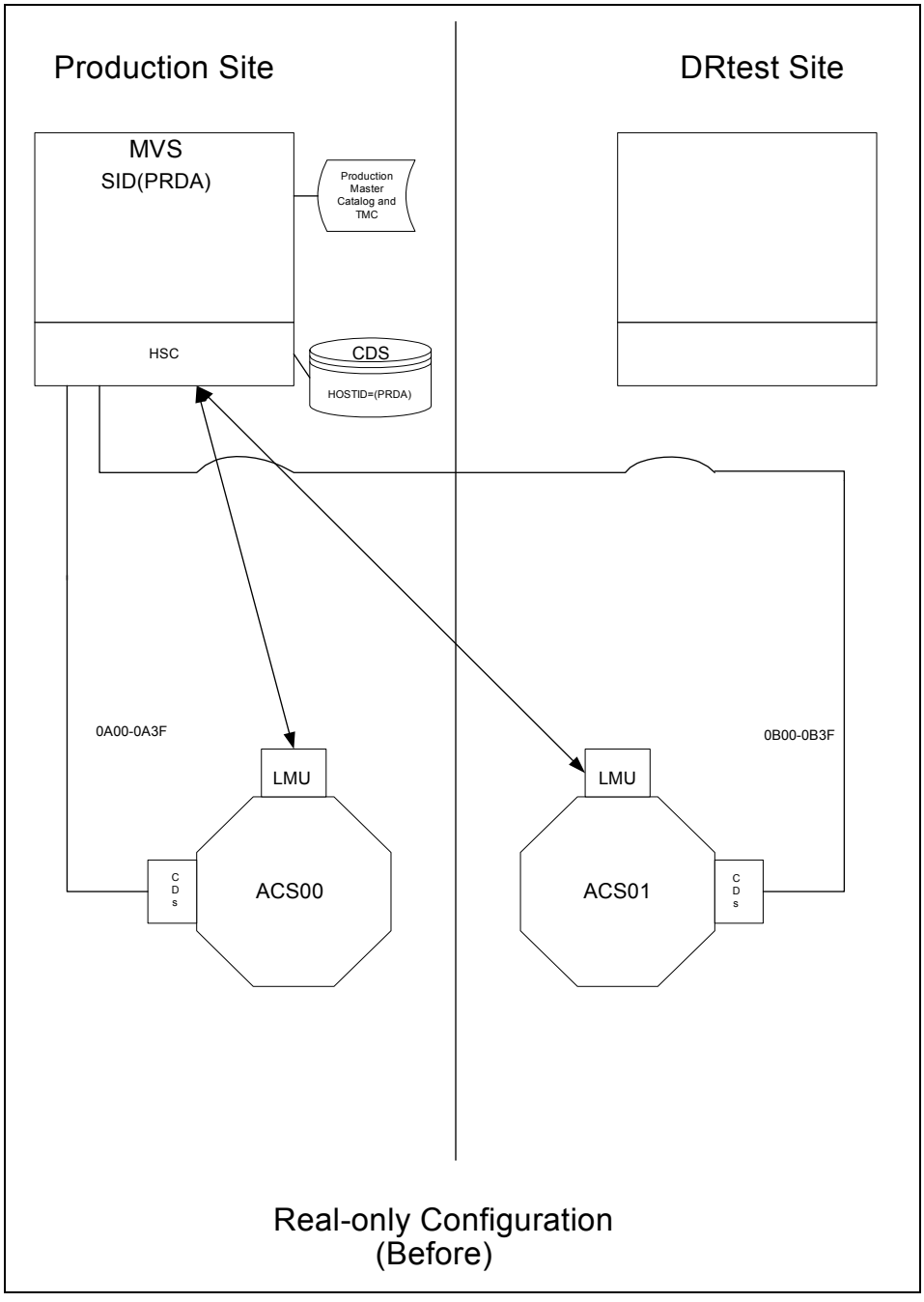

FIGURE 7-5 Real-Only Configuration - Before Running the DRTEST Utility

FIGURE 7-6 shows the system for Scenario 3 shows the system for Scenario 3 (real-only configuration) after running the DRTEST utility.

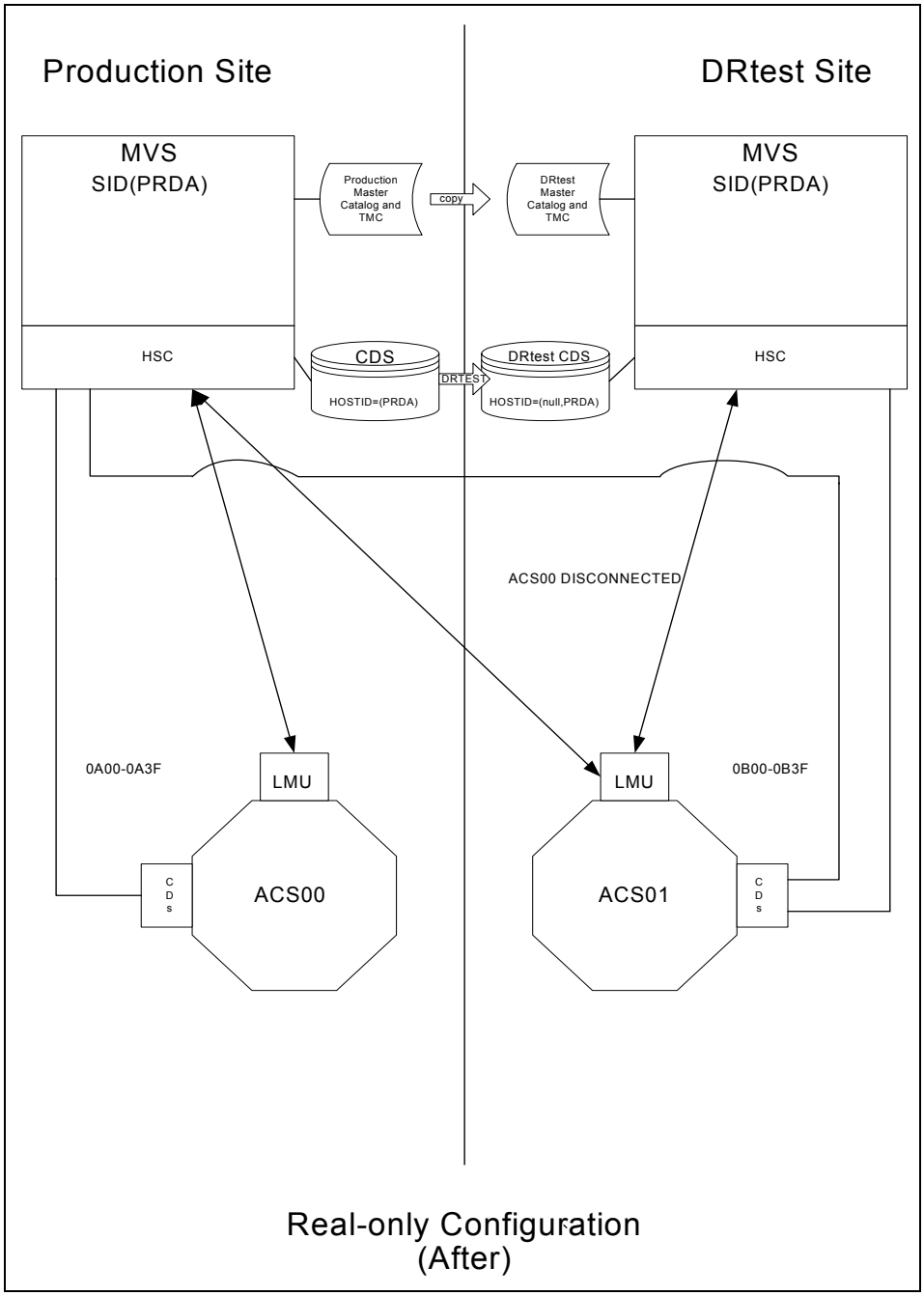

FIGURE 7-6 Real-Only Configuration - After Running the DRTEST Utility

Example JCL for Scenario 3

CODE EXAMPLE 7-12 Scenario 3, CREATE Step Only, PRIMEPRD Run Previously

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
DRACS(01) HOST(MVS1,MVS2)
```

Scenario 4: Clustered VTSSs with Production and DR Test Sites

As shown in FIGURE 7-7 on page 127, in normal operations, Scenario 4 is a Clustered VTSS configuration used for DR, with Production and DR Test sites cross-connected to the Production and DR Test ACSs. At the Production Site, `VTSS0` is the Primary, and `VTSS1` is the Secondary at the DR Test Site.

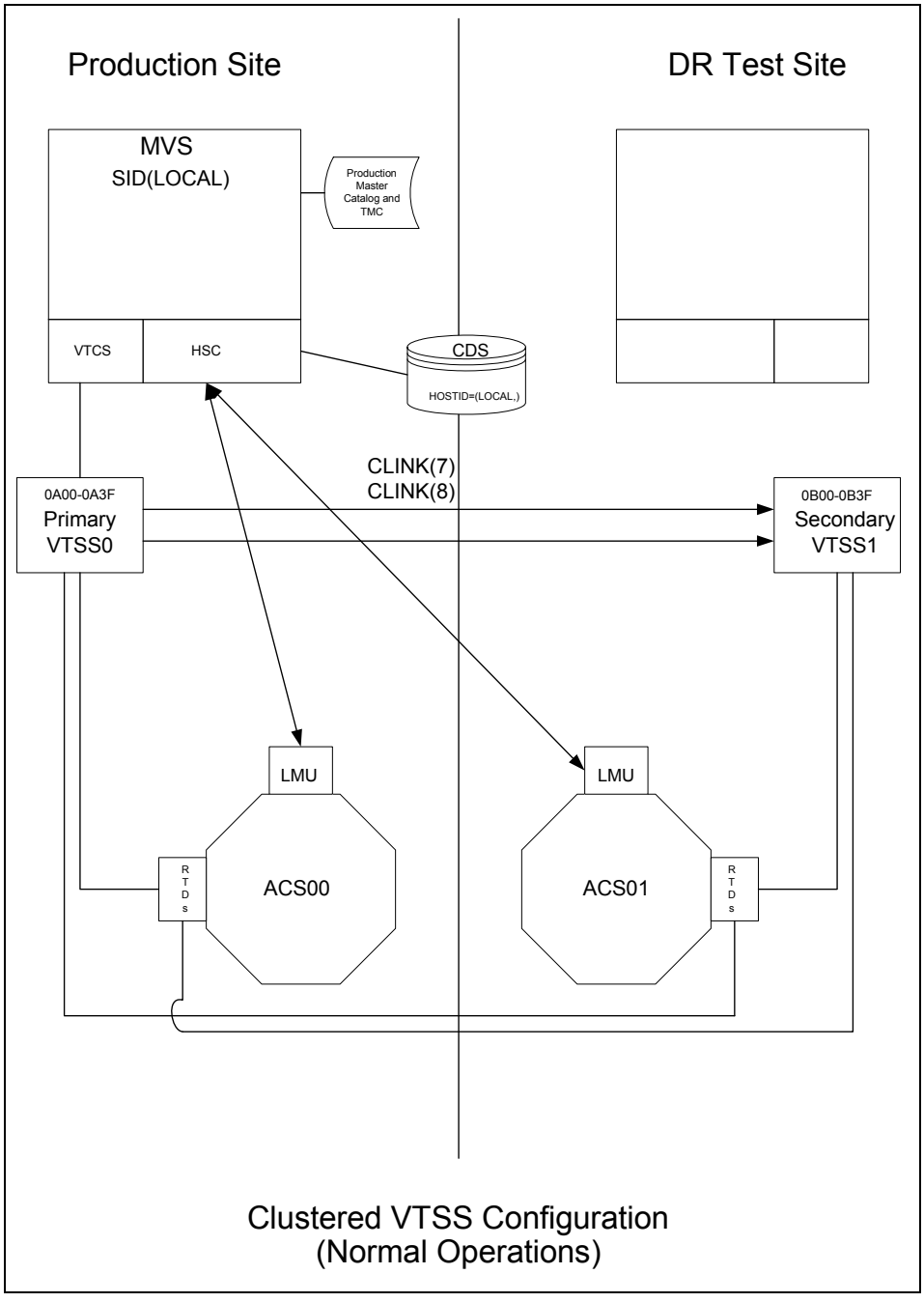

FIGURE 7-7 Primary/Secondary Clustered VTSS Configuration - Normal Operations

Example JCL for Scenario 4

CODE EXAMPLE 7-13 Scenario 4, CREATE Step Only, PRIMEPRD Run Previously

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
DRACS(01) DRVTSS(VTSS1) SHARE HOST(MVS1,MVS2)
```

.

What if you wanted to use the DR Test Site for testing? FIGURE 7-8 shows the system for Scenario 4 during the DR test.

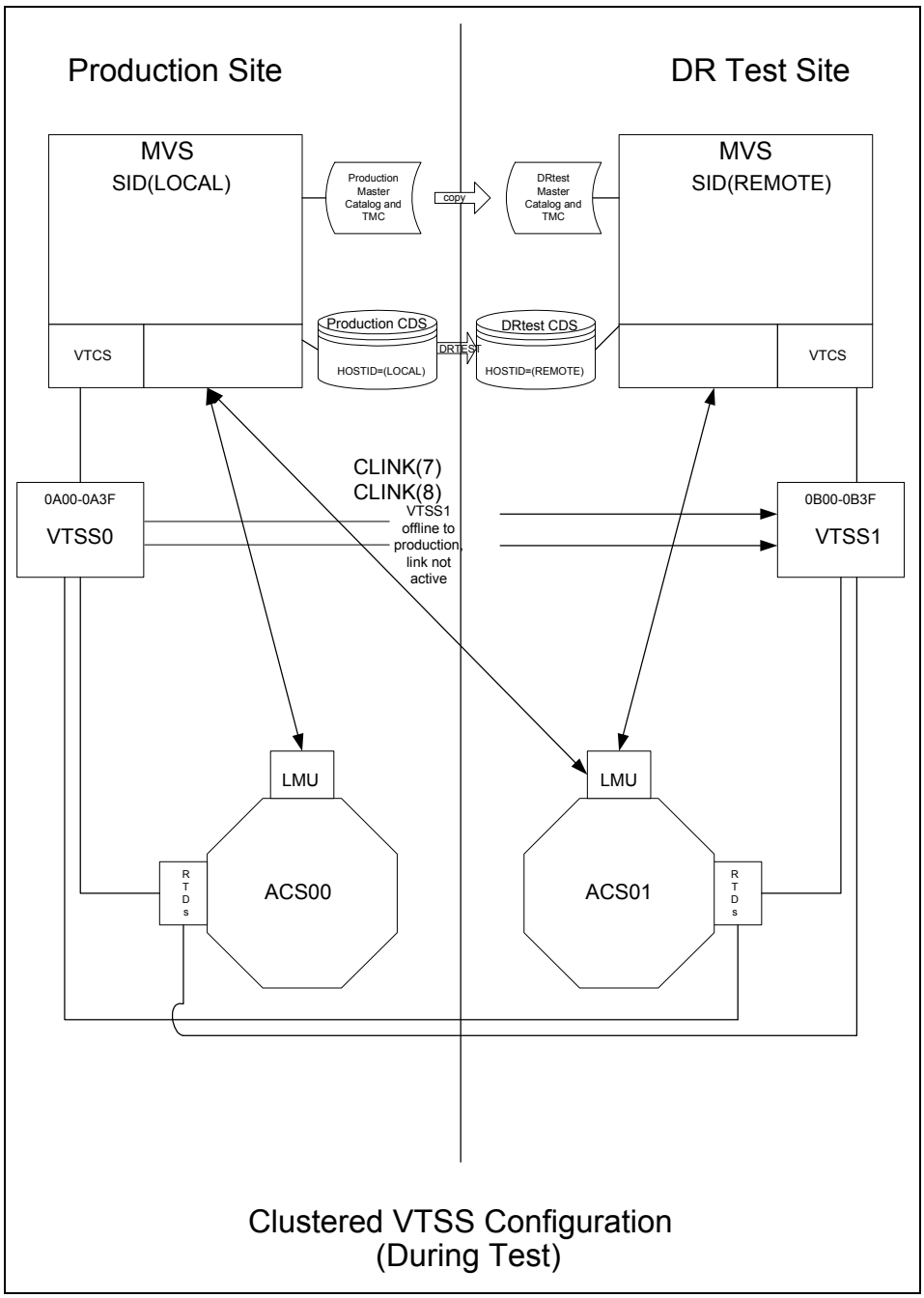

FIGURE 7-8 Primary/Secondary Clustered VTSS Configuration - During Test

Scenario 5: Production and Test Sites, ACS and VLE at Each Site

In Scenario 5, there is a single ACS at both the production and test sites, but no “spare” VTSS(s) at the test site used for testing. In normal operations, the production site writes and accesses VTVs on VTSSs at both sites, and output VTVs are always migrated immediately and duplexed, one copy to an MVC in the ACS, one copy to a VMVC in the VLE. In this configuration, you must demand migrate to zero one or more VTSSs at the test site and vary these VTSSs offline to the production system, so testing can take over the required VTSS resources. In addition, one or more LPARs at the test site function as displaced production systems, running in parallel with the actual production systems. **Both ACSs and VLEs are online to the production system.**

FIGURE 7-9 on page 131 shows the system for Scenario 5 before running the `DRTEST` utility.

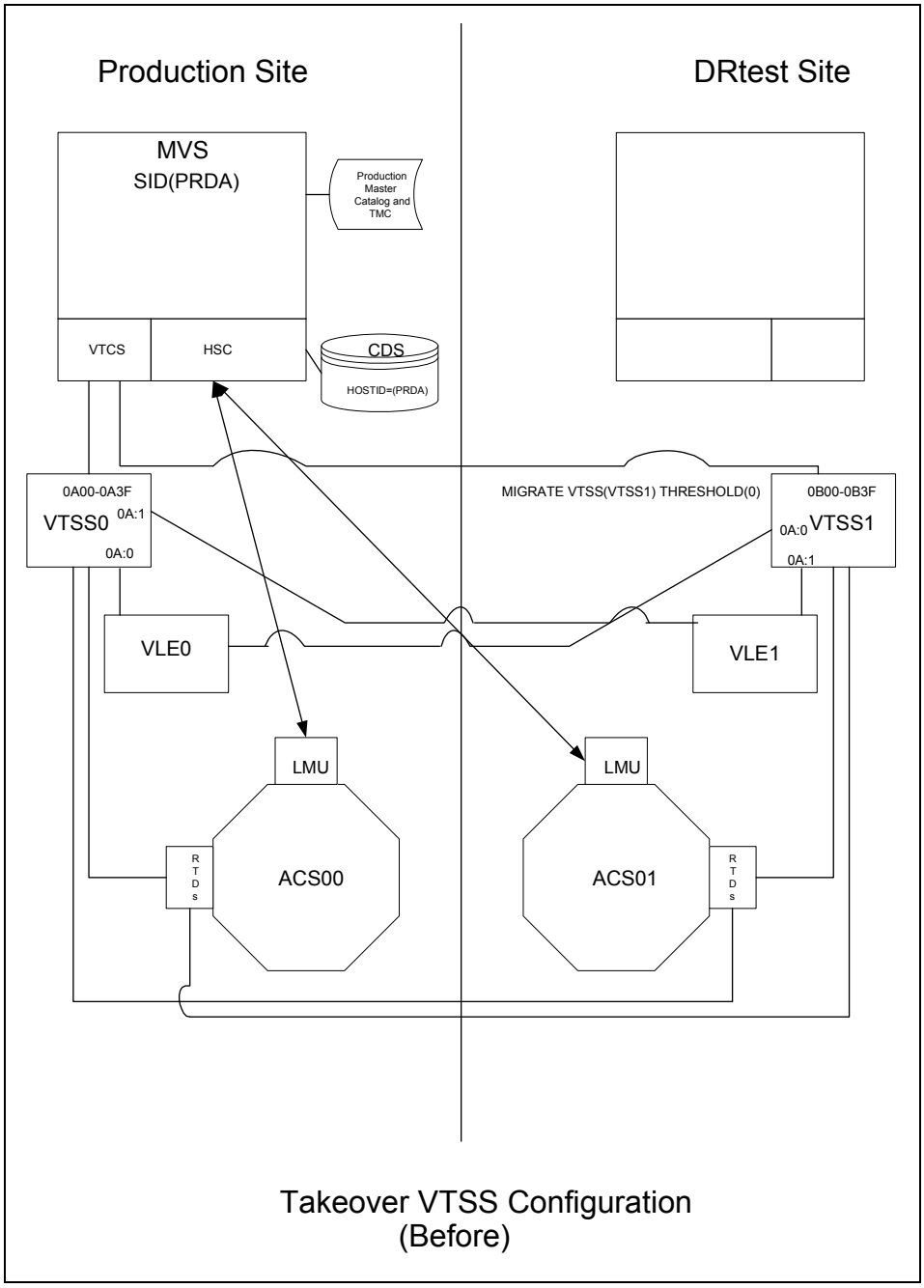

FIGURE 7-9 VLE and ACS Configuration - Before Running the DRTEST Utility

Example JCL for Scenario 5

CODE EXAMPLE 7-14 Scenario 5, CREATE Step Only, PRIMEPRD Run Previously

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
DRACS(01) DRVTSS(VTSS1) HOST(MVS1,MVS2) STORMNGR(VLE1)
```

FIGURE 7-2 shows the system for Scenario 5 after running the DRTEST utility.

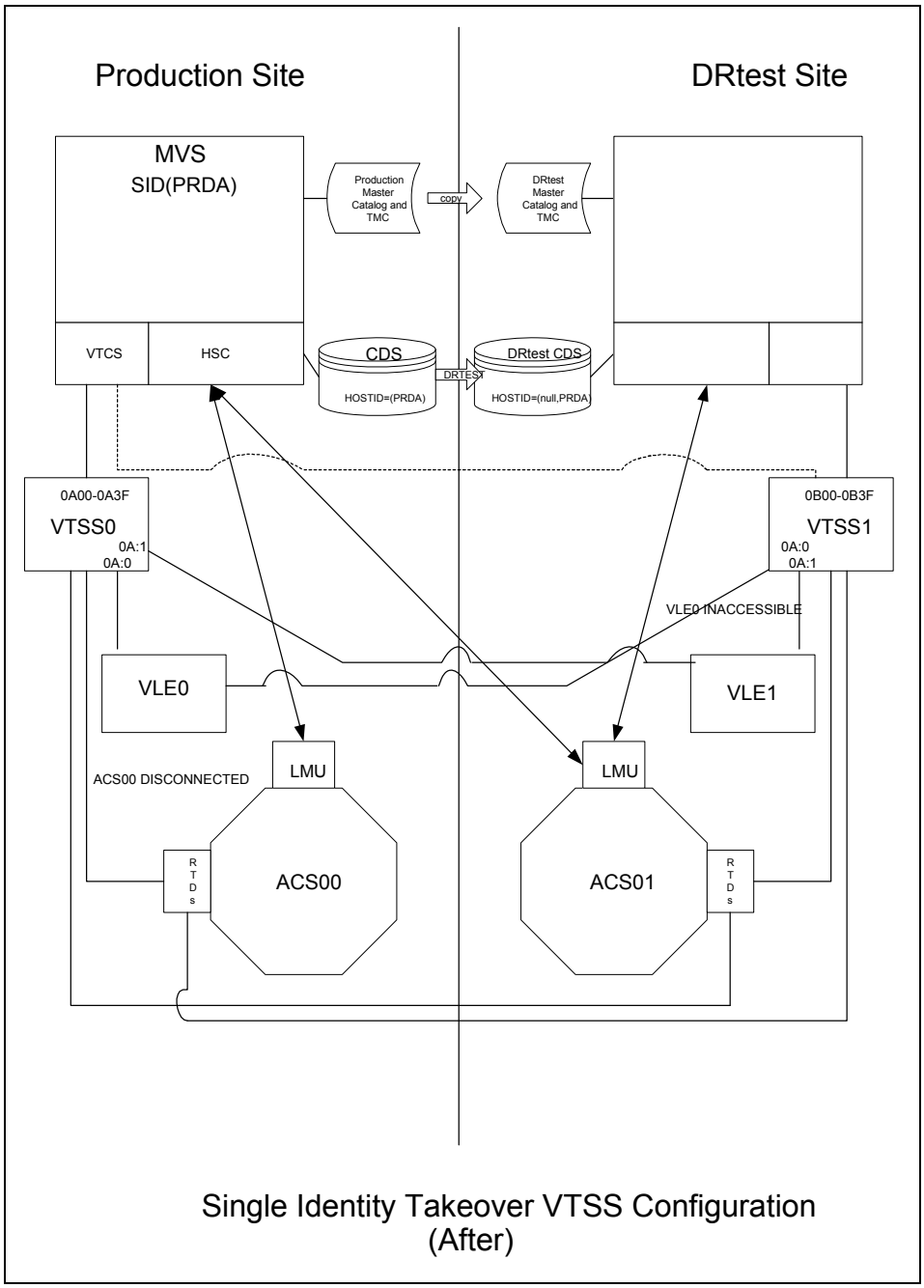

FIGURE 7-10 VLE and ACS Configuration - After Running the DRTEST Utility

Scenario 6: Production and Test Sites, VLE Only at Each Site

In Scenario 6, there is a single VTSS with a VLE attached at each site. The VTSS at the test site is **not** a spare and is used by the production site during normal operations. The output VTVs are always migrated immediately and duplexed to separate VMVCs, one in each VLE.

In this configuration, you must demand migrate to zero one or more VTSSs at the test site and vary these VTSSs offline to the production system, so testing can take over the required VTSS resources. In addition, one or more LPARs at the test site function as displaced production systems, running in parallel with the actual production systems. Both VLEs are online to the production system.

FIGURE 7-11 on page 135 shows the system for Scenario 6 before running the `DRTEST` utility.

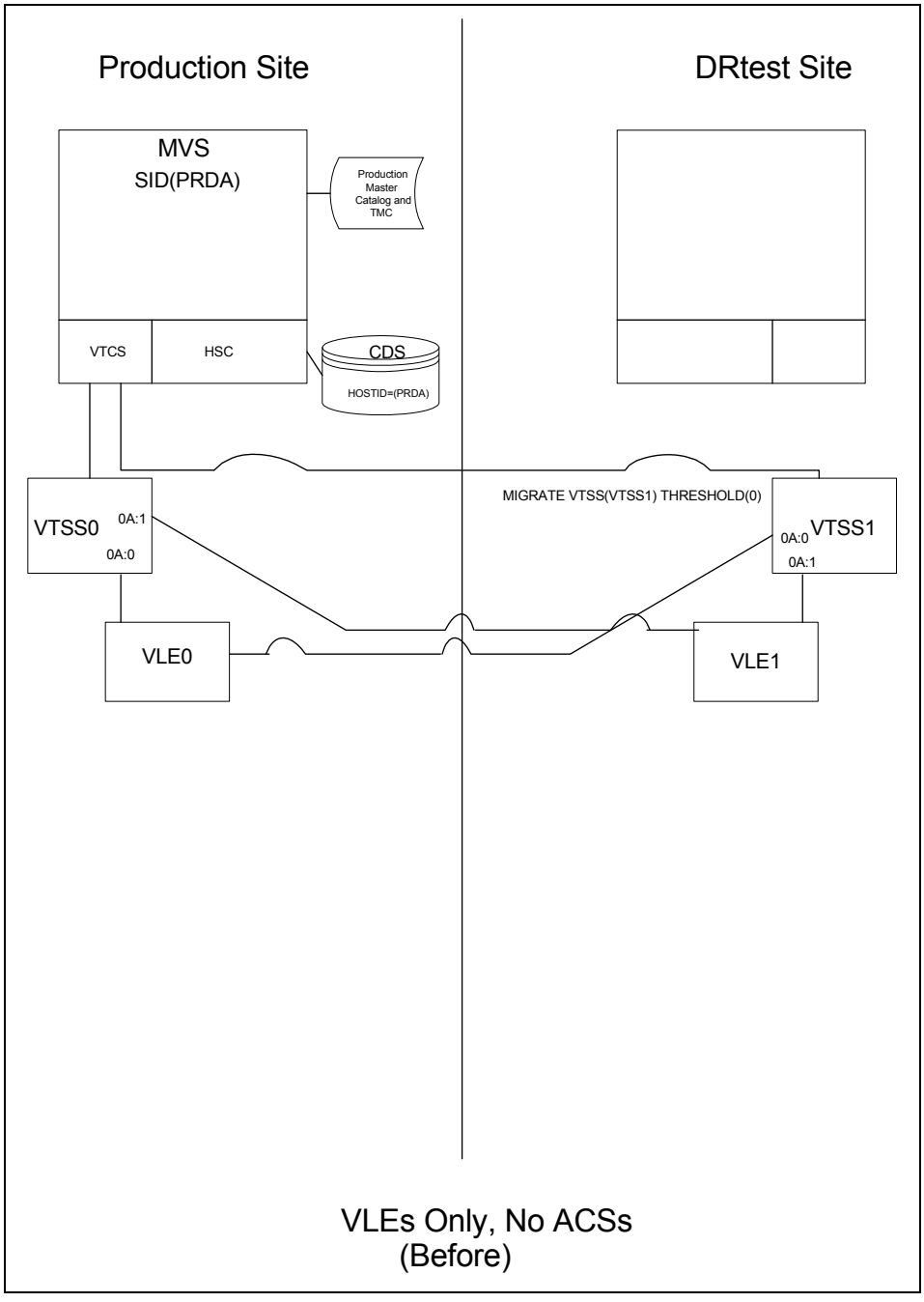

FIGURE 7-11 VLE Only Configuration - Before Running the DRTEST Utility

Example JCL for Scenario 6

CODE EXAMPLE 7-15 Scenario 6, CREATE Step Only, PRIMEPRD Run Previously

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
STORMNGR(VLE1) DRVTSS(VTSS1) HOST(MVS1,MVS2)
```

FIGURE 7-4 shows the system for Scenario 6 after running the DRTEST utility.

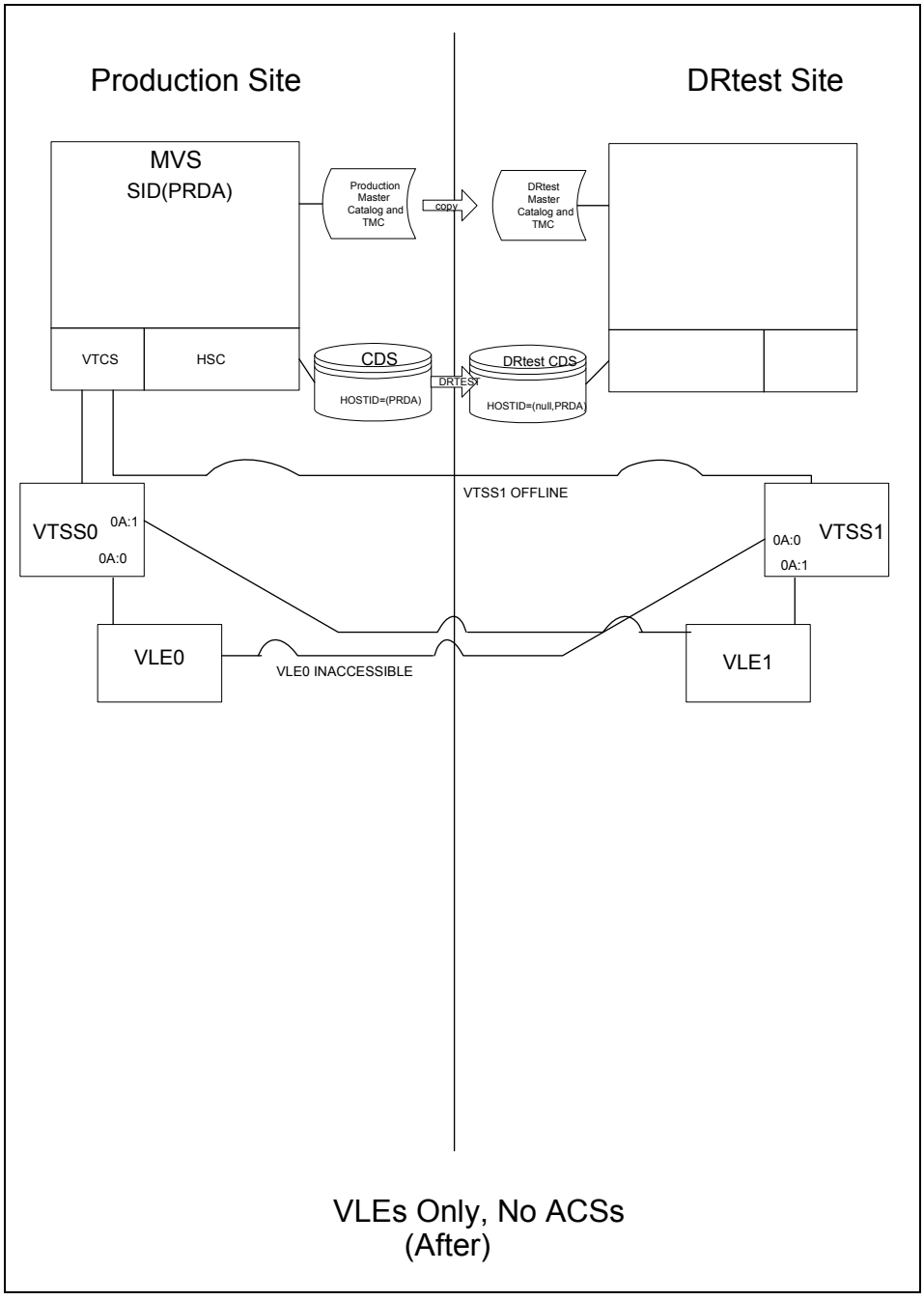

FIGURE 7-12 VLE Only Scenario- After Running the DRTEST Utility

Scenario 7: Clustered VTSSs (Tapeless) with Production and DR Test Sites

As shown in FIGURE 7-13, in normal operations, Scenario 7 is a Clustered VTSS (tapeless) configuration used for DR, with Production and DR Test site. At the Production Site, `VTSS0` is the Primary, and `VTSS1` is the Secondary at the DR Test Site.

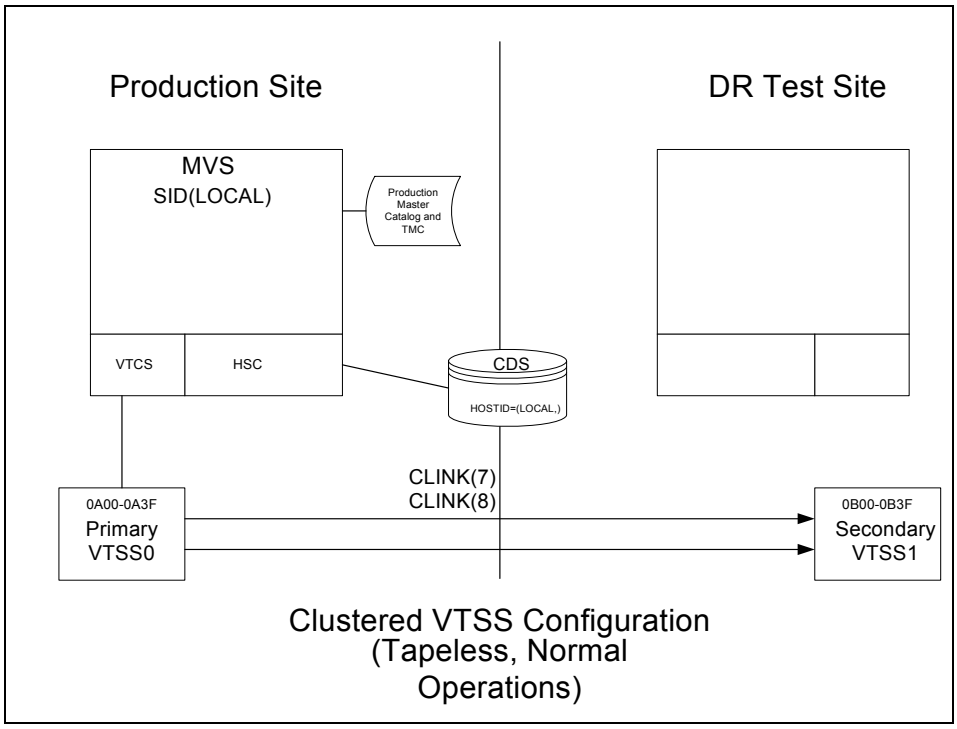

FIGURE 7-13 Primary/Secondary Clustered VTSS Configuration (Tapeless) - Before Running the DRTEST Utility

Example JCL for Scenario 7

CODE EXAMPLE 7-16 Scenario 7, CREATE Step Only, PRIMEPRD Run Previously

```
//DRTCREAT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSNEW1  DD DSN=hlq.DRTEST.SLSCNTL,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW2  DD DSN=hlq.DRTEST.SLSCNTL2,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSNEW3  DD DSN=hlq.DRTEST.SLSSTBY,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,x)
//SLSIN DD *
DRTEST CREATE NOUPDPRD +
DRVTSS(VTSS1) SHARE HOST(MVS1,MVS2))
```

FIGURE 7-14 Primary/Secondary Clustered VTSS Configuration - Tapeless, Normal Operation

What if you wanted to use the DR Test Site for testing? FIGURE 7-8 shows the system for Scenario 7 during the DR test.

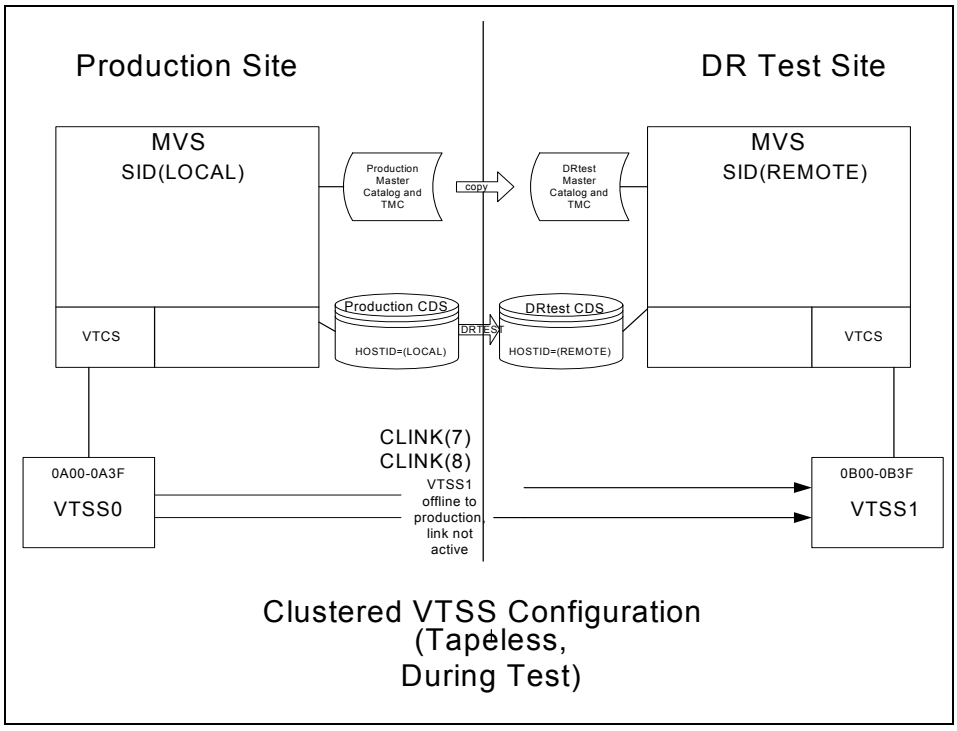

FIGURE 7-15 Primary/Secondary Clustered VTSS Configuration - Tapeless, During Test

CHAPTER 8

Creating System Recovery Points in VSM Environments

As described in "Defining the Recovery Point Objective (RPO)" on page 3, one of the keys to a successful DR solution is the ability to establish system checkpoints that ensure a consistent set of data can be used as a DR baseline.

For VSM environments, a valid DR baseline is where:

- All business critical data is secured at the designated DR location.
- A secure copy of the metadata (CDS, MVS Catalog, TMC) has been captured.
- The metadata copy is guaranteed to be valid when a disaster is declared (real or test).

VTCS provides the capability to create a DR baseline via the following functions:

- The DRMONitr utility monitors and ensures critical DR data reaches its designated recovery location. It allows job stream processing to stall awaiting the data to reach its destination. Once all data is accounted for, the utility ends. The DRMONitr utility can be run as a job step. On completion of the job step it is guaranteed that all monitored data has been accounted for and secured at the designated DR location.
- The DRCHKPT utility is used to ensure that data accessed through the CDS metadata remains valid for a set period. This guarantees that a CDS backup remains valid for a set period of time and, therefore, can be used to restore a VSM system back to a DR baseline. The DRCHKPT utility sets a date / time stamp in the active CDS which establishes the recovery point from which MVC content can be recovered from. Beginning at this recovery point in time, data content is guaranteed for some period of time in the future. Without the DRCHKPT utility, a CDS backup cannot be used to restore back to a DR baseline as elements in the CDS (VTV position on an MVC) may no longer be valid.

For more information, see *ELS Command, Control Statement, and Utility Reference*.

Also note the following:

- For VMVCs, MVCDRAIN with the EJECT parameter physically deletes the VTVs.

Caution – If you use the DRCHKPT utility and/or the CONFIG GLOBAL PROTECT parameter to protect CDS backup content for VMVCs, specifying MVCDR EJECT invalidates the CDS backup's VMVC content.

- For both VMVCs and MVCs, MVCDRAIN without the EJECT parameter does not delete the VTVs, but updates the CDS record to show no VTVs on the VMVC/MVC.

For more information, see *ELS Command, Control Statement, and Utility Reference*.

Checkpointing Examples

Example 1: Local MVC copies and Remote MVC copies

In this example:

- We employ the DRMONitr and DRCHKPT utilities to ensure the DR data has reached its recovery location and that the associated metadata (CDS backup) can be used to retrieve the VTV data if necessary.
- We have a local site with a VTSS plus an ACS (ACS 00), and a remote site with just an ACS (ACS 01) as shown in FIGURE 8-1 on page 143.

The example is a simple DR strategy where on a daily basis, copies of critical data are secured on the remote site along with the metadata. The remote VTV copies are the designated DR copies.

After the production jobs have completed a job is scheduled that:

- Monitors the remote copies for completion (DRMONitr).
- Checkpoints the CDS (DRCHKPT).
- Takes a backup of the metadata (CDS,TMC,MVS catalog) and secures on the remote site. **Note that** the metadata backups are key to the DR, it is assumed these are taken to a "well known" location or that their location is noted at a secure location.

This provides a daily synchronised DR checkpoint. In the case of a DR being declared the tape environment is restored back to the checkpoint and jobs are re-run from this known state.

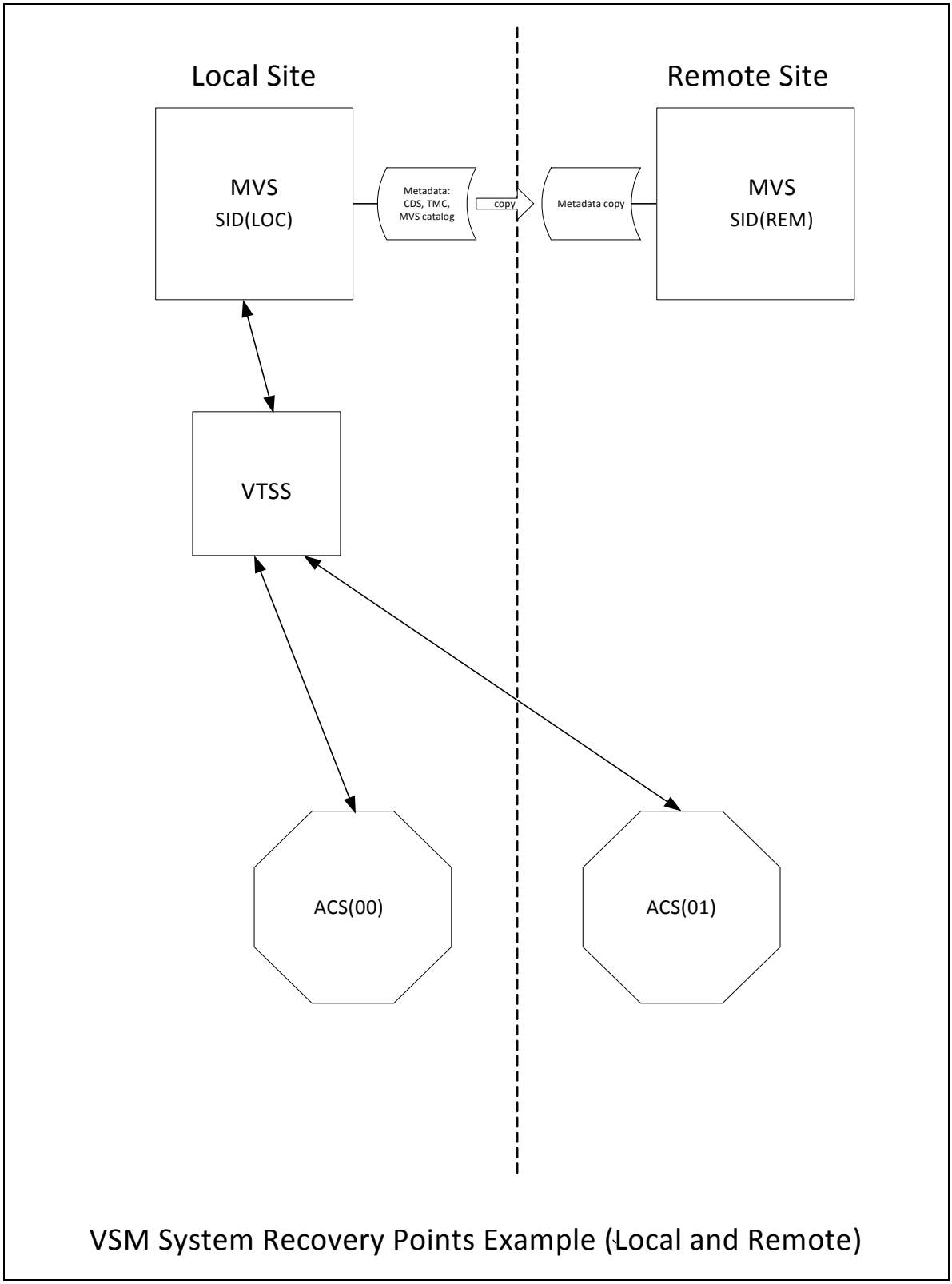

FIGURE 8-1 VSM System Recovery Points Example (Local and Remote)

To run this example using the configuration shown in FIGURE 8-1 on page 143**:**

1. **Create the following policy statements.**

```
MGMT NAME(DR)  MIGPOL(LOCAL,REMOTE) IMMDELAY(0)
STOR NAME(LOCAL) ACS(00)
STOR NAME(REMOTE) ACS(01)
```

Note – For an effective DR environment, you may also want to consider using MIGRSEL and MIGRVTV statements, which can be used to ensure DR copies are secured as early as possible.

---

2. **To ensure the critical data is secured in the remote location the following example** DRMONitr **jobstep is run.**

```
//MONITOR EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SYSIN         DD UNIT=SYSDA,SPACE=(TRK,1)
//*
//SYSPRINT  DD  SYSOUT=*
//SLSPRINT   DD SYSOUT=*
//SLSIN        DD *
DRMON MGMT(DR) STOR(REMOTE) MAXAGE(24) TIMEOUT(30)
```

In this example the DRMONitr utility will wait until all VTV copies, of management class DR less than 24 hours old, are delivered to the remote ACS. The utility is set to abort if the run time (or wait time) exceeds 30 minutes.

3. **Once all VTV copies have been delivered to the remote ACS, signaled by RC zero, the** DRCHKPT **runs to set the recovery point as shown in the following example.**

```
//CHKPT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SLSPRINT   DD SYSOUT=*
//SLSIN        DD *
DRCHKPT SET
```

In this example the DRCHKPT utility sets a time stamp, or recovery point, in the active CDS. Beginning at this recovery point in time, MVC copy content is guaranteed for a period of time in the future. (for example, until another CHKPT utility is run).

4. **Once the recovery point is set in the active CDS, a CDS backup should be taken immediately as shown in the following example.**

```
//CHKPT EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB  DD DSN=hlq.SEALINK,DISP=SHR
//SYSIN         DD UNIT=SYSDA,SPACE=(TRK,1)
//*
//SLSCNTL   DD  DSN=SHR,DSN=hlq.DBSEPRM
//SLSBKUP  DD  DSN=SHR,hlq.DBSEPRM.BKUP
//SYSPRINT  DD  SYSOUT=*
//SLSPRINT   DD SYSOUT=*
//SLSIN        DD *
BACKUP OPTION(COPY)
```

After the backup is taken, the MVC content, or metadata, is guaranteed to be valid for some point in time in the future (until a subsequent recovery or check point is set).

This completes this procedure. In the event of a DR declaration (the local production site is no longer available) then either:

- The MVCs and all other critical data (metadata copies for example) are transported to another facility where a mirror of the production Local site is available

OR

- A replica of the production Local site is constructed at the remote location.

The metadata is restored (CDS, TMC, MVS Catalog). On restarting the tape environment everything can proceed (roll forward) from the DR sync point.

Example 2: Using CONFIG RECLAIM PROTECT

In this example, we back up the CDS every 24 hours. We must ensure that MVC content, or CDS metadata, within the CDS backup remains valid until the subsequent CDS backup is taken.

This example shows MVC protection set as 28 hours. For more information on the CONFIG RECLAIM PROTECT parameter, see *ELS Command, Control Statement, and Utility Reference*.

1. **Set CONFIG GLOBAL PROTECT = 28.**
2. **Day 1, back up the CDS.**
   - Any MVCs drained/reclaimed after this backup cannot be overwritten for 28 hours.
   - **Day 1 CDS backup** is now the recovery point until the next CDS backup.
3. **Day 2, back up the CDS.**
   - Any MVCs drained/reclaimed after this backup cannot be overwritten for 28 hours.
   - **Day 2CDS backup** is now the recovery point until the next CDS backup.

CHAPTER 9

Using VLE for Disaster Recovery

The use of the VLE (Virtual Library Extension) as a disaster recovery solution provides a simplified and non-disruptive method of performing DR testing, as well as recovery from a business disruption event.

The VLE is managed by the system like a library (ACS). However, because the VLE uses disk storage rather than tape storage, and because it maintains an internal inventory of VTVs in its contents, it offers features that a real library cannot provide:

- The VLE is a "tapeless" solution, avoiding issues of media management.
- Data is sent to the VLE using IP, and does not require channel extension.
- The VLE can perform an MVC audit in a matter of a few seconds, using its internal database, compared to mounting and reading an MVC cartridge.

This chapter describes the use of the VLE in a simple two site environment. However, it should be noted that the solution supports any number of sites with any number of VLEs at each site. Also, one of the sites can be a DR-only site, not running MVS LPARs except during a DR test or declared disaster.

The detailed procedure that follows use the following environment:

There are two sites, SITE1 and SITE2. Each site has one VSM and one VLE. In this example SITE2 is described as a DR-only site, but SITE2 can also be a production site defined as a mirror image of SITE1.

Note – The size of the VLE buffer at SITE2 must be sufficient to hold both the migrated production data as well as the data created during a DR test.

Normal Production Mode

During normal production policies are defined at SITE1 to migrate one copy of the data to the local VLE at SITE1 and a second copy to the remote VLE at SITE2. Additional copies can be created if desired, including both copies in another VLE as well as tape copies.

The following shows an example of policies defined at SITE1:

SMC Definitions are used to assign a MGMTCLAS name of VLEPROD to data sets with a high level qualifier of "PAYROLL."

```
POLICY NAME(VLEPOL) MEDIA(VIRTUAL) MGMT(VLEMGMT) + SUBP(VIRTSCR)

TAPEREQ DSN(PAYROLL.*) POLICY(VLEPOL)
```

HSC POOLPARM/VOLPARM definitions are used to define the production volumes:

```
POOLPARM TYPE(MVC)  NAME(LOCAL)

VOLPARM VOLSER(VLL000-VLL099)

POOLPARM TYPE(MVC)  NAME(VAULT1)

VOLPARM VOLSER(VLV000-VLV099)

POOLPARM TYPE(SCRATCH) NAME(VIRTSCR)

VOLPARM VOLSER(V00000-V99999) MEDIA(VIRTUAL)
```

Note that the MVCs in the pools LOCAL and VAULT1 are VMVCs (virtual MVCs) in the SITE1 and SITE2 VLEs, respectively, and do not have a media type associated with them.

VTCS STORCLAS and MGMTCLAS are used to define the VTCS policies:

```
STOR NAME(VLE1) STORMNGR(SITE1VLE) MVCPOOL(LOCAL)

STOR NAME(VLE2) STORMNGR(SITE2VLE) MVCPOOL(VAULT1)

MGMT NAME(VLEMGMT) DELSCR(YES) MIGPOL(VLE1,VLE2)
```

When a job runs with a data set starting with the high level qualifier "PAYROLL," SMC uses the TAPEREQ and POLICY to assign a MGMTCLAS of VLEPROD to the mount request. VTCS selects a virtual scratch volume in the pool LOCSCR (range V00000-V99999) and assigns it a MGMTCLAS of VLEPROD. After the volume is dismounted, one copy is migrated to the local VLE (STORMNGR SITE1VLE) and the second copy is migrated to the remote VLE (STORMNGR SITE2VLE).

Running a DR Test with VLE

The setup process for a DR test at SITE2 is simple and fast, and requires minimal restrictions at SITE1.

The basic steps are:

1. Create a new CDS at SITE2 containing only basic configuration data.
2. Mark the SITE1 VMVCs as READONLY to avoid conflicts.
3. Perform an audit of the virtual production MVCs in the SITE2VLE. This step populates the CDS with existing virtual metadata. Depending on the number of VTVs in the VLE this step may take anywhere from a few minutes to less than an hour.
4. Run the DR test workload, using a range of VTVs and MVCs that does not overlap with the production volumes.

The remainder of this section gives the details of defining the parameters at the DR site, and describes steps you must take to ensure that the contents production VMVCs are not changed during the test.

1. Creating the DR test CDS.
   a. Use the LIBGEN/SLICREAT process to create the CDS at the SITE2. Note that you create this CDS even if you are already running production work at SITE2. The new CDS contains only DR data from SITE1. Also note that you must define at least one ACS in the LIBGEN macros, even if your configuration does not contain physical tape.
   b. Run the SET VOLPARM utility to define the volumes for the DR test:

```
POOLPARM TYPE(MVC)  NAME(VAULT1)
VOLPARM VOLSER(VLV000-VLV099)
POOLPARM TYPE(EXTERNAL) NAME(PRODVTVS)
VOLPARM VOLSER(V00000-V99999) MEDIA(VIRTUAL)
POOLPARM TYPE(MVC)  NAME(DRMVC)
VOLPARM VOLSER(VLT000-VLT099)
POOLPARM TYPE(SCRATCH) NAME(VIRTSCR)
VOLPARM VOLSER(VT0000-VT9999) MEDIA(VIRTUAL)
```

   Note that the first two pools define volumes created by SITE1 that will be used as input to the test at SITE2. The pool type of EXTERNAL indicates that these are volumes that are not part of a scratch subpool. The last two pools are local pools that will be used as output from the test at SITE2.

c. Define VTCS MGMTCLAS and STORCLAS that will be used for the DR test:

```
STOR NAME(DRVLE) STORMNGR(SITE2VLE) MVCPOOL(DRMVC)
MGMT NAME(VLEMGMT) DELSCR(YES) MIGPOL(DRVLE)
```

d. Note that because the MGMTCLAS and scratch subpools in the SITE2 DR system have the same names as the production policies (but different definitions), you can now use the same SMC POLICY and TAPEREQ statements for your SITE2 DR test as you use in your SITE1 production.

e. Bring up HSC/VTCS on the DR test LPAR.

2. Mark the production MVCs as READONLY.

a. This is a critical step in the process and must be done on both the production CDS at SITE1 and the DR test CDS at SITE2. Note that once the MVCs have been defined as READONLY on the production CDS, you can continue to run normal processing, including:

i. RECLAIM. Automatic reclaim will not select an MVC in READONLY status

ii. SCRATCH. Although VTVs will get updated in the production CDS to be in scratch status, and may be reused, the copy on the VLE read only virtual MVC is unaffected.

iii. Normal processing to append to or overwrite the VTVs on the VMVCs. The new VTV versions will be migrated to new VMVCs, while the copy on the VLE read only virtual MVC is unaffected.

Note that you cannot, however, run the DRAIN utility against these MVCs, as that removes the VLE copy of the virtual MVC metadata.

b. Use the utility function ACTMVCGN to select the production MVCs at the production site, using the production CDS. This utility generates control statements to set the READONLY flag on the MVCs it selects, as well as control statements to turn off the READONLY flag after the test is complete. Using the ALL keyword on the ACTMVCGN control statement ensures the full MVCs are selected for READONLY processing, which allows automatic reclaim to execute on the production system without impact to the DR test. The SLUSMAUD DD statement should also be included to generate AUDIT statements for the VMVCs that will be used in the test. Note that you can, if you wish, run the ACTMVCGN utility at the production site to create the production updates, and at the DR site on a mirrored copy of the CDS to create the DR test CDS updates. Following is an example of the JCL to run this utility:

```
//ACTMVCGN JOB (ACCT),'ACTMVCGN',NOTIFY=&SYSUID
//ACTMVCG1  EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB        DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT      DD SYSOUT=*
//*      NOTE: CDS DD statements are optional if running at the production site with an
//*      active HSC LPAR.
//SLSCNTL       DD DSN=hlq.DBASEPRM,DISP=SHR
//SLSCNTL2     DD DSN=hlq.DBASESEC,DISP=SHR
//SLSSTBY       DD DSN=hlq.DBASESBY,DISP=SHR
//*                        NOTE: MVCMAINT READONLY(ON) STATEMENTS
//SLUSMVON  DD DSN=hlq.SLUSMVON,DISP=(NEW,CATLG,DELETE),
//                    SPACE=(CYL,1)
//*                        NOTE: MVCMAINT READONLY(OFF) STATEMENTS
//SLUSMVOF   DD DSN=hlq.SLUSMVOF,DISP=(NEW,CATLG,DELETE),
//                    SPACE=(CYL,1)
//*      NOTE: AUDIT MVC(VVVVVV) STATEMENTS
//SLUSMAUD   DD DSN=hlq.SLUSMAUD,DISP=(NEW,CATLG,DELETE),
//                    SPACE=(CYL,1)
//*                        NOTE: THE FOLLOWING SELECTS ALL "NON-EMPTY" VMVCS
//SLSIN           DD *
 ACTMVCGN ALL MVCPOOL(VAULT1)
/*
```

3. At the production site, run the MVCMAINT utility function to mark the VMVCs as READONLY.

```
//RDONLYON   EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB      DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT     DD SYSOUT=*
//*                  NOTE: EXEC MVCMAINT TO SET READONLY(ON).  Output of
//*     ACTMVCGN utility.
//SLSIN          DD DSN=hlq.SLUSMVON,DISP=SHR
```

4. Bring up the HSC/VTCS at the DR site.
5. Run an MVC audit of the production VMVCs in the SITE2 VLE using the newly created SITE2 CDS and the output of the ACTMVCGN utility. This step populates the CDS metadata containing the relationships between the VTVs and the VMVCs.

```
//AUDIT EXEC PGM=SLUADMIN
//STEPLIB            DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT           DD SYSOUT=*
//*    NOTE: AUDIT CONTROL STATEMENTS FROM ACTMVCGN UTILITY
//SLSIN              DD   DSN=hlq.SLUSMAUD,DISP=SHR
```

   a. Optionally, you can recall VTVs that will be used in the DR test into the VTSS buffer, using either LCM or other method of selecting VTVs to recall. However, this step is not necessary, since recalls from the VLE buffer are relatively fast.

6. Run the MVCMAINT utility using the output of the ACTMVCGN READONLY(ON) to set the production VMVCs to READONLY at SITE2 on the DR CDS.

```
//RDONLYON   EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB      DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT     DD SYSOUT=*
//*                  NOTE: EXEC MVCMAINT TO SET READONLY(ON).  Output of
//*     ACTMVCGN utility.
//SLSIN          DD DSN=hlq.SLUSMVON,DISP=SHR
```

7. Optional: Prior to starting your DR test, you may want to run a VTVRPT and an MVCRPT to validate the contents of your DR test CDS.

8. Run your DR test workload.

   a. Bring up SMC. If you used the same names on your MGMTCLAS and scratch subpools as the production system, you can use your production TAPEREQ and POLICY statements. It is recommended but not required that you use a different TapePlex name for the DR test TapePlex.

   b. Run your DR test workload using the SMC and new HSC/VTCS CDS.

   c. There are no limitations on updating production VTV volumes during the DR test. Data on production VTVs may be appended to (DISP=MOD) or overwritten (DISP=OLD). These updates do not affect the contents of the VTV copy on the READONLY production virtual MVC, and therefore do not affect the production copy of the data.

Cleanup After a DR Test with VLE

When the DR test is complete, the purpose of the cleanup is to remove metadata from the VTSS and from the VLE so that the next DR test will not see this data. Note that the DR test HSC/VTCS must remain active until the cleanup is complete. The steps are:

1. Run the SCRATCH utility function to scratch all VTVs created during the test from both the VTSS and from the VLE DR test VMVCs. When the DELSCR(YES) parameter is specified on the DR test MGMTCLAS, running the scratch utility causes the VTVs to be deleted from both the buffer and from the VLE metadata.

```
//SCRATCH EXEC PGM=SLUADMIN
//STEPLIB          DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT         DD SYSOUT=*
//SLSIN            DD *
SCRATCH VOL(VT0000-VT9999)
```

   Note that if you have modified any production VTVs using DISP=MOD or DISP=OLD, these VTVs remain in the buffer and on the VLE.

   By scratching the VTVs in the DR test subpool after the test, you minimize the amount of time required to clean up the VTSS, and minimize the amount of data left in the VLE after the completion of the test.

2. Migrate the VTSS to 0.

```
//MIGRTO0   EXEC PGM=SLUADMIN
//STEPLIB          DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT         DD SYSOUT=*
//SLSIN            DD *
MIGRATE VTSS(DRVTSS) THRESHLD(0)
```

This step is required only if the output of your DR test included new versions of production VTVs

3. Verify that the DR VTSS is now empty.

```
//AUDVTSS    EXEC PGM=SLUADMIN
//STEPLIB          DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT         DD SYSOUT=*
//SLSIN            DD *
AUDIT VTSS(DRVTSS)
```

   Note that if you have modified production VTVs during your DR test, copies of this data, as well as the metadata, remain in the VLE for the DR test MVC pool (VLT000-VLT099, VTVs V00000-V99999). During the next DR test, these VMVCs will be written starting at the logical beginning of tape, and any data they contain will be removed from the VLE. Since the new DR test CDS has no knowledge of this data, it will not affect the next DR test.

4. At the production site, use the READONLY(OFF) control cards created by the ACTMVCGN utility at the beginning of the test to put the production VMVCs back into a writable status.

```
//RDONLYOF   EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB      DD DSN=hlq.SEALINK,DISP=SHR
//SLSPRINT     DD SYSOUT=*
//*                  NOTE: EXEC MVCMAINT TO SET READONLY(OFF)
//SLSIN        DD DSN=hlq.SLUSMVOF,DISP=SHR
```

Using VLE for Business Continuance

When an outage occurs at SITE1 that requires SITE2 to take over SITE1's workload, the process is almost identical to the DR test procedure.

If a DR test happens to be executing when the SITE1 outage occurs, follow the above process to clean up after the DR test and stop the DR test.

In order to begin running the SITE1 workload at SITE2, follow the procedure described above for starting a DR test. You will, of course, omit the step of marking the production VMVCs as READONLY on the production CDS, as there is no "production" CDS to update. However, you will use the mirrored copy of the production CDS to generate the MVCMAINT READONLY control cards for the production MVCs in the VLE.

You will also want to use the DR test policies that segregate the VTVs being created and the output VMVCs into a separate range, to avoid any possibility of corrupting the production data until after the business continuance has been verified.

NOTE: If production jobs performing DISP=MOD processing for tape data do not have a defined synchronization point, it is possible that the contents of a VTV at the time of an outage may be unpredictable. StorageTek recommends that all disaster recovery procedures be reviewed to ensure predictable synchronization points for tape data.

APPENDIX **A**

Clustered VTSS Examples

"Using Clustered VTSS Configurations" on page 75 provides the basics of VTSS Clustering and this appendix provides the following Clustering examples:

Uni-Directional Clustered VTSS

FIGURE A-1 shows an example of a Uni-Directional Clustered VTSS Dual ACS system. **Note that** in this example, FICON ports provide the CLINK connections. In this example, I only have one MVS host, but it's putting out a lot of critical data that I want protected to the max using two brand new VSM4s that I just purchased. No problem: VTSS1 is the Primary VTSS, and it's connected to the Secondary (VTSS2) via Cluster Links (CLINKs). If the Management Class for a VTV specifies replication, presto, when the VTV arrives in VTSS1, it is *replicated* (copied) to VTSS2, and also immediately migrated (with KEEP). Result: I have increased data availability (there's a copy of the VTV in each VTSS in case one fails) *and* data protection (the VTV is also on square tape in case both VTSSs go offline). Clustered VTSS is a great solution, therefore, for business continuity and business resumption.

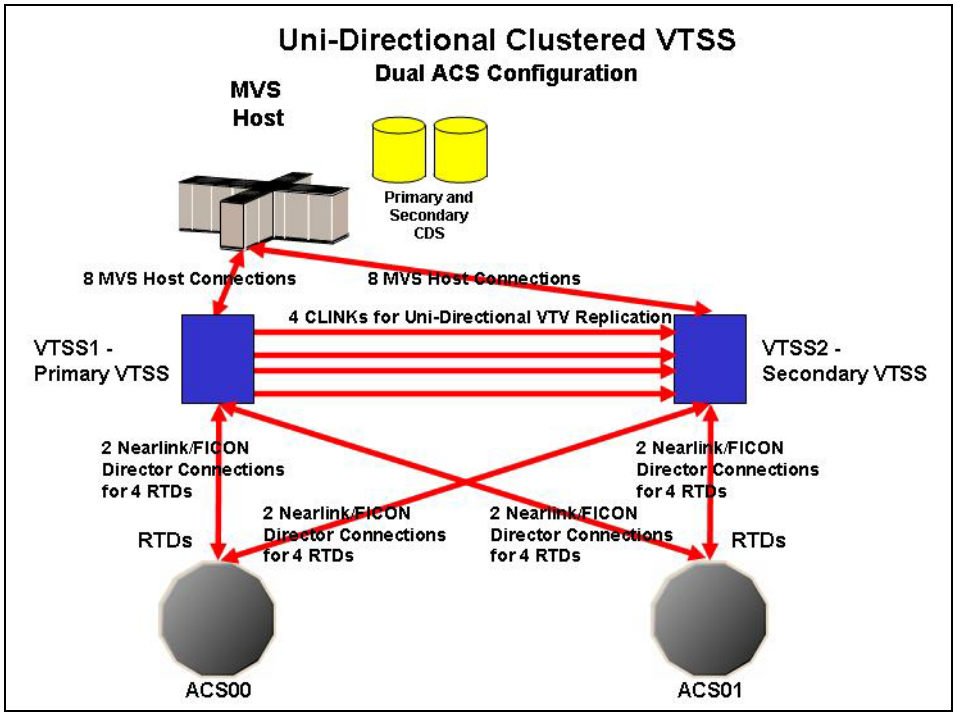

FIGURE A-1 Dual ACS Uni-Directional Clustered VTSS Configuration

Now it's time to take a look at the hardware for this Clustered configuration. FIGURE A-2 shows CONFIG channel interface identifiers for a VSM4 with 8 VCF cards. In this configuration, we've allocated:

- 8 Host ports.
- 4 ports for RTDs. The RTD ports are all connected to FICON directors, each of which is attached to RTDs, so the CHANIF identifiers for both RTDs are shown on each port. This allows Back-End connection to 8 RTDs, although only one RTD per port/Director can be active at a time.

- 4 ports for CLINK connections to form a Uni-Directional VTSS Cluster, and 8 ports to host connections. To form the clustered VTSS, we'll have two VSM4s (VTSS1 and VTSS2) configured identically as shown in FIGURE A-2.

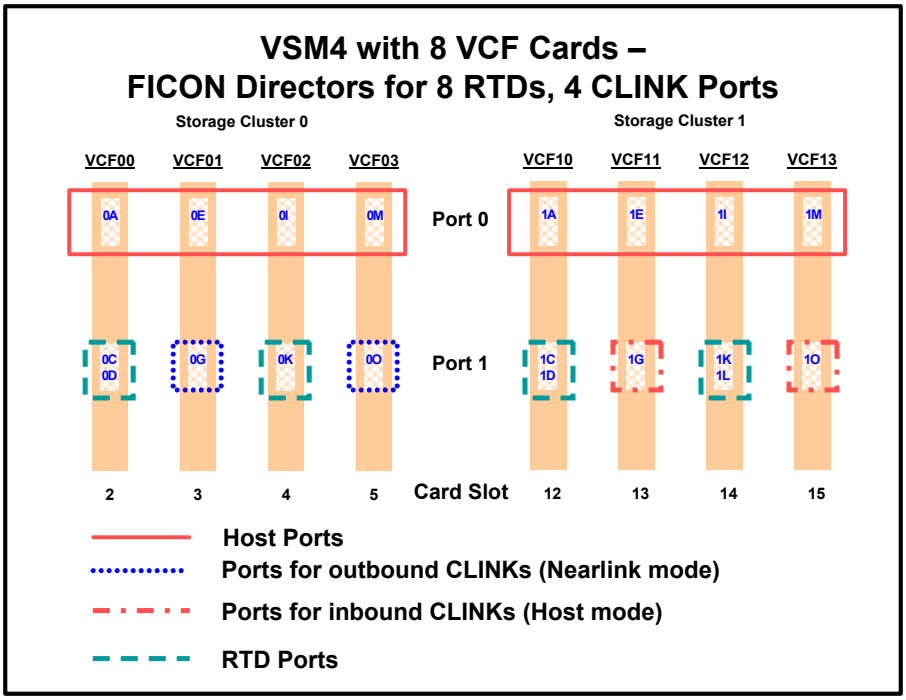

FIGURE A-2 VSM4 with 8 VCF Cards, 8 Host Ports, FICON Directors for 8 RTDs, 4 CLINK Ports

Okay, we've seen what our example Uni-Directional Cluster looks like, and we've seen the VCF card port configurations required. Now let's tie it all together in "Configuring and Managing a Uni-Directional Clustered VTSS System" on page 160.

Configuring and Managing a Uni-Directional Clustered VTSS System

To configure and manage the Uni-Directional Clustered system shown in Figure 11. on page 26, do the following:

1. **Ensure that your system has the Clustered VTSS requirements.**
2. **Use CONFIG to create CLUSTER and CLINK statements to define the VTSS Cluster and its connections.**

   FIGURE A-3 shows example CONFIG JCL to define a Uni-Directional Cluster of two VSM4s (VTSS1 and VTSS2) as shown in Figure 11 on page 26. **Note that**:

   - The CLUSTER statement defines the Cluster as consisting of VTSS1 and VTSS2.
   - There are CLINK statements using the sending (Nearlink Mode) ports of **only VTSS1** to enable the Cluster as Uni-Directional, where VTSS1 is the Primary and VTSS2 is the Secondary.

```
//CREATECFGEXEC PGM=SWSADMIN,PARM='MIXED'
//STEPLIBDD DSN=hlq.SLSLINK,DISP=SHR
//SLSCNTLDD DSN=FEDB.VSMLMULT.DBASEPRM,DISP=SHR
//SLSCNTL2DD DSN=FEDB.VSMLMULT.DBASESEC,DISP=SHR
//SLSSTBYDD DSN=FEDB.VSMLMULT.DBASETBY,DISP=SHR
//CFG22202DD DSN=FEDB.VSMLMULT.CFG22202,DISP=SHR
//SLSPRINTDD   SYSOUT=*
//SLSINDD   *
 CONFIG RESET CDSLEVEL(V62ABOVE)
 GLOBAL MAXVTV=65000 MVCFREE=60 VTVATTR=SCRATCH RECALWER=YES
 LOCKSTR=STK_VTCS_LOCKS VTVPAGE=LARGE
 RECLAIM THRESHLD=70 MAXMVC=30 START=40 CONMVC=5
 VTVVOL LOW=905000 HIGH=999999 SCRATCH
 VTVVOL LOW=C00000 HIGH=C25000 SCRATCH
 VTVVOL LOW=RMM000 HIGH=RMM020 SCRATCH
 MVCVOL LOW=N25980 HIGH=N25989
 MVCVOL LOW=N35000 HIGH=N35999
 VTSS NAME=VTSS1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
 RTD  NAME=PR11A00 DEVNO=1A00 CHANIF=0C
 RTD  NAME=PR11A01 DEVNO=1A01 CHANIF=0D
 RTD  NAME=PR11A02 DEVNO=1A02 CHANIF=0K
 RTD  NAME=PR11A03 DEVNO=1A03 CHANIF=0L
 RTD  NAME=PR12A08 DEVNO=2A08 CHANIF=1C
 RTD  NAME=PR12A09 DEVNO=2A09 CHANIF=1D
 RTD  NAME=PR12A0A DEVNO=2A0A CHANIF=1K
 RTD  NAME=PR12A0B DEVNO=2A0B CHANIF=1L
 VTD LOW=9900 HIGH=99FF
 VTSS NAME=VTSS2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
 RTD  NAME=PR23A00 DEVNO=3A00 CHANIF=0C
 RTD  NAME=PR23A01 DEVNO=3A01 CHANIF=0D
 RTD  NAME=PR23A02 DEVNO=3A02 CHANIF=0K
 RTD  NAME=PR23A03 DEVNO=3A03 CHANIF=0L
 RTD  NAME=PR24A08 DEVNO=4A08 CHANIF=1C
 RTD  NAME=PR24A09 DEVNO=4A09 CHANIF=1D
 RTD  NAME=PR24A0A DEVNO=4A0A CHANIF=1K
 RTD  NAME=PR24A0B DEVNO=4A0B CHANIF=1L
 VTD LOW=9900 HIGH=99FF
 CLUSTER NAME=CLUSTER1 VTSSs(VTSS1,VTSS2)
 CLINK VTSS=VTSS1 CHANIF=0G
 CLINK VTSS=VTSS1 CHANIF=0O
 CLINK VTSS=VTSS1 CHANIF=1G
 CLINK VTSS=VTSS1 CHANIF=1O
```

FIGURE A-3 CONFIG example: Dual ACS Uni-Directional Clustered VTSS System, VSM4 FICON Back-End.

3. **Specify the Conditional Replication setting on the CONFIG GLOBAL statement.**

```
CONFIG GLOBAL REPLICAT=CHANGED
```

FIGURE A-4 CONFIG GLOBAL Setting for VTV Replication

In FIGURE A-5, CONFIG GLOBAL REPLICAT=CHANGED specifies:

- Replicate VTVs only if the VTV is updated and an identical copy does not exist in the Secondary.
- Via the MIGPOL parameter, migrate duplexed to ACSs 00 and 01 by Storage Classes you will create in Step 5 on page 31.

What if I wanted to unconditionally replicate VTVs? I would specify (you guessed it), CONFIG GLOBAL REPLICAT=ALWAYS.

4. **Create a Management Class that specifies VTV replication and two Storage Classes to migrate (duplexed) the replicated VTVs.**

```
MGMT NAME(VSMREPL) REPLICAT(YES) MIGPOL(REPLSTR1,REPLSTR2)
```

FIGURE A-5 Management Class for VTV Replication

Note –

- Note the subtle interaction between GLOBAL REPLICAT, which specifies *when* the replication can occur, and MGMTclas REPLICAT(YES), which says, "when the GLOBAL REPLICAT condition says it's time, go ahead and replicate."
- The Management Class VSMREPL **does not** specify an immediate migrate policy. VTV replication automatically enforces immediate migrate. The VTVs in this Management Class will be added to the immediate migration queue on VTSS once the replication has completed. Note that duplexing **is not** a requirement for replicate VTVs. For more information, see "How Clustered VTSS Configurations Work" on page 40.

5. **Create the Storage Classes for the MVCs that contain the replicated, migrated VTVs.**

```
STOR NAME(REPLSTR1) ACS(00) MEDIA(STK1R) MIRATE(RECEIVER)
STOR NAME(REPLSTR2) ACS(01) MEDIA(STK1R) MIGRATE(RECEIVER)
```

FIGURE A-6 Storage Classes for Replicated, Migrated VTVs

In FIGURE A-6, the STORclas statement defines Storage Classes REPLSTR1 and REPLSTR2 referenced in the MIGPOL parameter in Step 4 on page 30. **Also note** that the MIGRATE parameters on the Storage Classes specify that the VTSS receiving the replicated VTV...in this case VTSS2, the Secondary, does the migration to both ACSs. This is a handy way of ensuring that the Secondary functions as the "migrate engine."

6. **Load the MGMTclas and STORclas control statements with the MGMTDEF command.**

```
MGMTDEF DSN(hsc.parms)
```

FIGURE A-7 MGMTDEF Command to Load Statements

7. **Create a TAPEREQ statement to route the critical data to VSM and assign Management Class VSMREPL to the data.**

```
TAPEREQ DSN(*.PAYROLL.**) MEDIA(VIRTUAL) MGMT(VSMREPL)
```

FIGURE A-8 TAPEREQ Statement to Route Critical Data, Assign Management Class VSMREPL

In FIGURE A-8, the TAPEREQ statement specifies:

- Route data sets with HLQ mask *.PAYROLL.** to VSM...
- .....and assign Management Class VSMREPL that you created in Step 4 on page 30.

Caution – To replicate VTVs, **both VTSS1** and VTSS2 must be varied online to VTCS so that VTCS can send control commands to both VTSSs. See "How Clustered VTSS Configurations Work" on page 80 for more information.

Note – Also note the following:

- You can also use esoteric substitution via SMC TAPEREQ statement or SMC DFSMS ACS routines to route replication jobs to VSM. For more information, see *SMC Configuration and Administration Guide*.

8. **Check your HSC PARMLIB options to ensure that subtype 28 records are enabled.**

   If enabled, VTSS clustering writes a subtype 28 record for each replication performed.

   ...and you're home free!

Bi-Directional Clustered VTSS

FIGURE A-9 shows an example of a Bi-Directional Clustered VTSS Dual ACS system.**Note that** in this example, FICON ports provide the CLINK connections. This system is very similar to the uni-directional example, but goes one step further: There are two MVS hosts sharing a CDS, and everything in the picture is cross-connected. I basically have sites mirroring each other for the ultimate in data availability and protection. To make this happen...that is, to make it bi-directional...I have to configure the two VTSSs as peers via the CLINK statements.

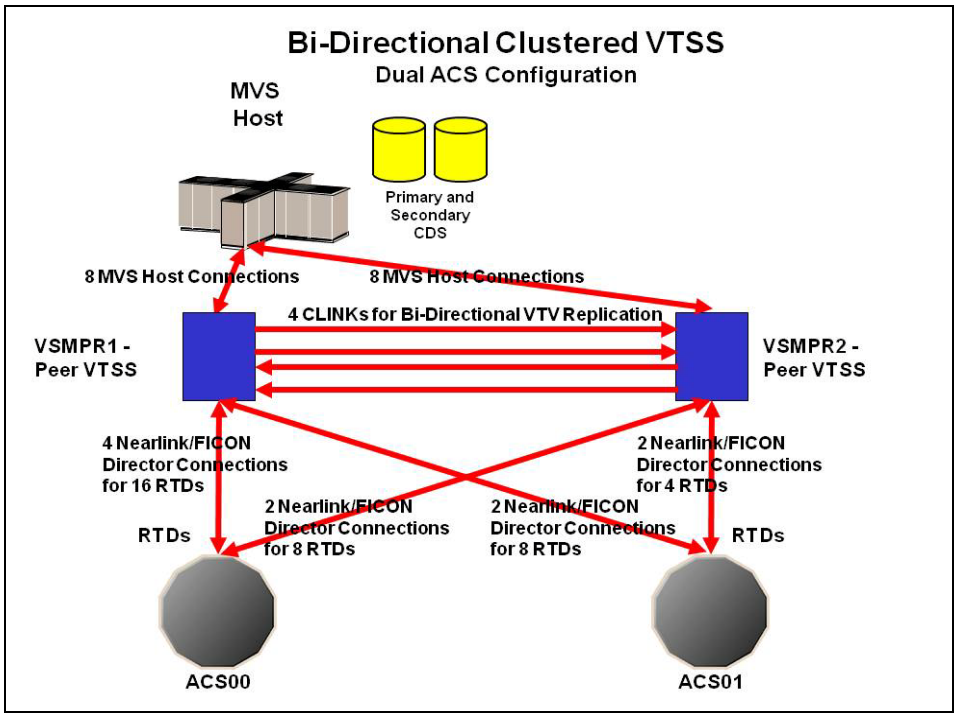

FIGURE A-9 Dual ACS Bi-Directional Clustered VTSS Configuration

Note –

- Bi-Directional Clustering **requires** VTCS 6.1 and above! You **cannot** configure a Bi-Directional Cluster at releases lower than VTCS 6.1!
- This configuration is shown with the feature that enables up to a total of 16 simultaneous NearLink I/O transfers, which can be spread across multiple targets on as many as 14 NearLink ports, and up to a total of 2 simultaneous NearLink I/O transfers per port. This feature requires VTSS microcode D02.06.00.00 or higher.

FIGURE A-10 shows CONFIG channel interface identifiers for VSMPR1 shown in FIGURE A-9 on page 164. In this configuration, we've allocated:

- 8 Host ports.
- 6 ports for RTDs. The RTD ports are all connected to FICON directors, each of which is attached to 4 RTDs, so the CHANIF identifiers for all 4 RTDs are shown on each port. This allows Back-End connection to 24 RTDs, although only one RTD per port/Director can be active at a time.
- 4 ports via FICON directors. Two are Nearlink for the originator, Two are host mode for the terminator for CLINK connections to form a Bi-Directional VTSS Cluster.

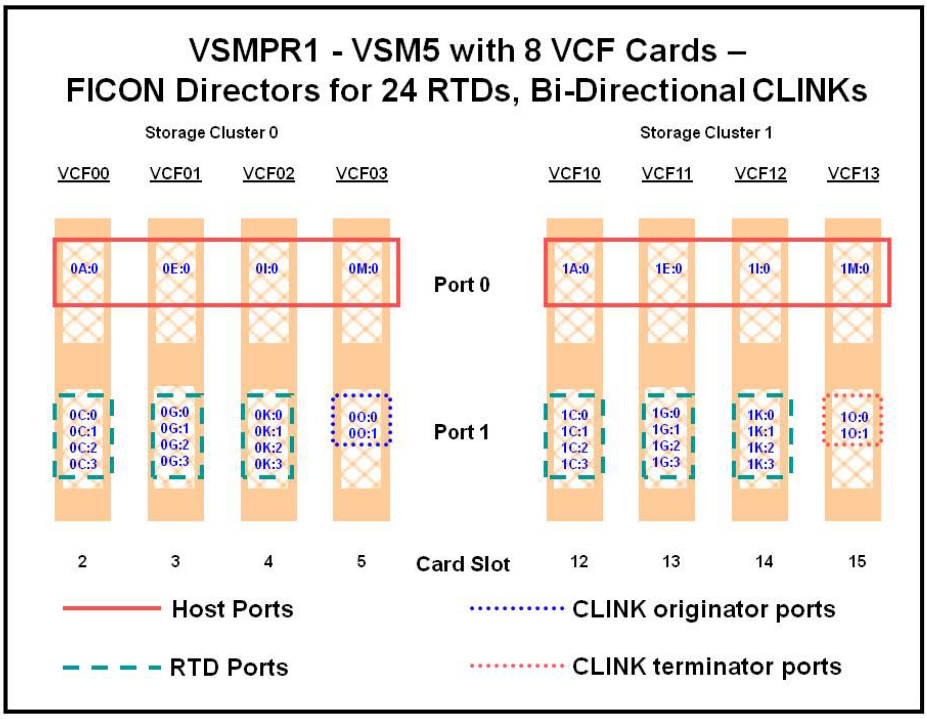

FIGURE A-10 VSMPR1 - VSM5 with 8 VCF Cards, 8 Host Ports, FICON Directors for 24 RTDs, 4 CLINKs

FIGURE A-11 shows CONFIG channel interface identifiers for a VSMPR1, a VSM5 in a Bi-directional Cluster with 8 VCF cards and the Maximum 32 RTDs feature enabled. In this configuration, we've allocated:

- 8 Host ports.
- 6 ports for RTDs. The RTD ports are all connected to FICON directors, each of which is attached to 4 RTDs, so the CHANIF identifiers for all 4 RTDs are shown on each port. This allows Back-End connection to 24 RTDs, although only one RTD per port/Director can be active at a time.
- 4 ports via FICON directors. Two are Nearlink for the originator, Two are host mode for the terminator for CLINK connections to form a Bi-Directional VTSS Cluster.

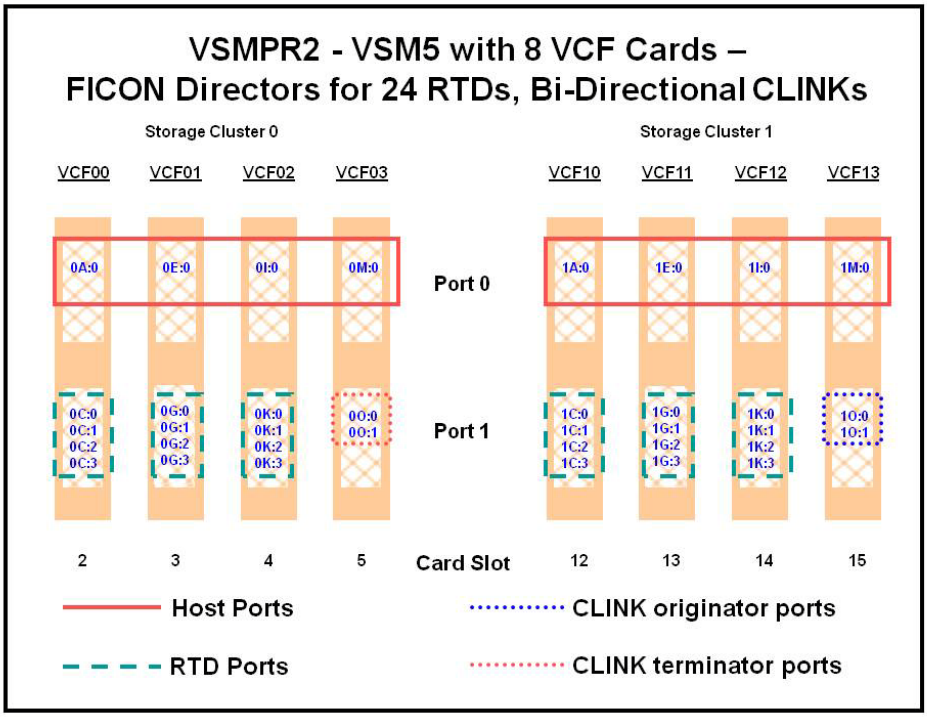

FIGURE A-11 VSMPR2 - VSM5 with 8 VCF Cards, 8 Host Ports, FICON Directors for 24 RTDs, 4 CLINKs

---

Caution – As shown in FIGURE 6-5 on page 88, each CLINK must be attached to **the same Storage Cluster** on each VTSS, **which is a requirement**. Failure to configure in this manner can produce Replicate, Channel, and Communication errors!
So as shown, the Nearlink ports (CLINK originators) on VSMPR1 are on Storage Cluster 0 and the Host ports (CLINK terminators) on VSMPR2 are also on Storage Cluster 0. The same is true for the CLINK connections for data flowing in the other direction; they are both on Storage Cluster 1.

Configuring and Managing a Bi-Directional Clustered System

To configure and manage the Bi-Directional Clustered system shown in FIGURE A-9 on page 164**, do the following:**

1. **Ensure that your system has the Clustered VTSS requirements described in *Installing ELS*.**
2. **Use CONFIG to create CLUSTER and CLINK statements to define the VTSS Cluster and its connections.**

   FIGURE A-12 shows example CONFIG JCL to define a Bi-Directional Cluster of two VSM4s (VSMPR1 and VSMPR2) as shown in Figure 11 on page 26. **Note that**:

   - The CLUSTER statement defines the Cluster as consisting of VSMPR1 and VSMPR2.

- There are CLINK statements using the sending (Nearlink Mode) ports of **both VTSSs** to enable the Cluster as Bi-Directional and they follow the rule of "connect using the same Storage Cluster on each VTSS for the sending and receiving ports of each CLINK".

```
//CREATECF EXEC PGM=SWSADMIN,PARM='MIXED'
//STEPLIB DD DSN=hlq.SLSLINK,DISP=SHR
//SLSCNTL DD DSN=FEDB.VSMLMULT.DBASEPRM,DISP=SHR
//SLSCNTL2 DD DSN=FEDB.VSMLMULT.DBASESEC,DISP=SHR
//SLSSTBY DD DSN=FEDB.VSMLMULT.DBASETBY,DISP=SHR
//SLSPRINT DD   SYSOUT=*
//SLSIN DD   *
 CONFIG RESET CDSLEVEL(V61ABOVE)
GLOBAL MAXVTV=32000 MVCFREE=40 VTVATTR=SCRATCH RECALWER=YES LOCKSTR=
VTCS_LOCKS
REPLICAT=ALWAYS VTVPAGE=LARGE SYNCHREP=YES MAXRTDS=32
RECLAIMTHRESHLD=70 MAXMVC=40  START=35
RECLAIMTHRESHLD=70MAXMVC=40  START=35
 VTVVOL LOW=905000 HIGH=999999 SCRATCH
 VTVVOL LOW=C00000 HIGH=C25000 SCRATCH
 VTVVOL LOW=RMM000 HIGH=RMM020 SCRATCH
 MVCVOL LOW=N25980 HIGH=N25989
 MVCVOL LOW=N35000 HIGH=N35999
 VTSS NAME=VSMPR1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD  NAME=VPR12A00 DEVNO=2A00 CHANIF=0C:0
 RTD  NAME=VPR12A01 DEVNO=2A01 CHANIF=0C:1
 RTD  NAME=VPR12A02 DEVNO=2A02 CHANIF=0C:2
 RTD  NAME=VPR12A03 DEVNO=2A03 CHANIF=0C:3
 RTD  NAME=VPR12A04 DEVNO=2A04 CHANIF=0G:0
 RTD  NAME=VPR12A05 DEVNO=2A05 CHANIF=0G:1
 RTD  NAME=VPR12A06 DEVNO=2A06 CHANIF=0G:2
 RTD  NAME=VPR12A07 DEVNO=2A07 CHANIF=0G:3
 RTD  NAME=VPR12A08 DEVNO=2A08 CHANIF=0K:0
 RTD  NAME=VPR12A09 DEVNO=2A09 CHANIF=0K:1
 RTD  NAME=VPR12A0A DEVNO=2A0A CHANIF=0K:2
 RTD  NAME=VPR12A0B DEVNO=2A0B CHANIF=0K:3
RTD  NAME=VPR13A00 DEVNO=3A00 CHANIF=1C:0
 RTD  NAME=VPR13A01 DEVNO=3A01 CHANIF=1C:1
RTD  NAME=VPR13A02 DEVNO=3A02 CHANIF=1C:2
 RTD  NAME=VPR13A03 DEVNO=3A03 CHANIF=1C:3
 RTD  NAME=VPR13A04 DEVNO=3A04 CHANIF=1G:0
 RTD  NAME=VPR13A05 DEVNO=3A05 CHANIF=1G:1
 RTD  NAME=VPR13A06 DEVNO=3A06 CHANIF=1G:2
 RTD  NAME=VPR13A07 DEVNO=3A07 CHANIF=1G:3
 RTD  NAME=VPR13A08 DEVNO=3A08 CHANIF=1K:0
 RTD  NAME=VPR13A09 DEVNO=3A09 CHANIF=1K:1
 RTD  NAME=VPR13A0A DEVNO=3A0A CHANIF=1K:2
 RTD  NAME=VPR13A0B DEVNO=3A0B CHANIF=1K:3
VTD LOW=9900 HIGH=99FF
```

FIGURE A-12 CONFIG example: Dual ACS Bi-Directional Clustered VTSS System (Part 1)

```
 VTSS NAME=VSMPR2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD  NAME=VPR22B00 DEVNO=2B00 CHANIF=0C:0
 RTD  NAME=VPR22B01 DEVNO=2B01 CHANIF=0C:1
 RTD  NAME=VPR22B02 DEVNO=2B02 CHANIF=0C:2
 RTD  NAME=VPR22B03 DEVNO=2B03 CHANIF=0C:3
 RTD  NAME=VPR22B04 DEVNO=2B04 CHANIF=0G:0
 RTD  NAME=VPR22B05 DEVNO=2B05 CHANIF=0G:1
 RTD  NAME=VPR22B06 DEVNO=2B06 CHANIF=0G:2
 RTD  NAME=VPR22B07 DEVNO=2B07 CHANIF=0G:3
 RTD  NAME=VPR22B08 DEVNO=2B08 CHANIF=0K:0
 RTD  NAME=VPR22B09 DEVNO=2B09 CHANIF=0K:1
 RTD  NAME=VPR22B0A DEVNO=2B0A CHANIF=0K:2
 RTD  NAME=VPR22B0B DEVNO=2B0B CHANIF=0K:3
RTD  NAME=VPR23B00 DEVNO=3B00 CHANIF=1C:0
 RTD  NAME=VPR23B01 DEVNO=3B01 CHANIF=1C:1
RTD  NAME=VPR23B02 DEVNO=3B02 CHANIF=1C:2
 RTD  NAME=VPR23B03 DEVNO=3B03 CHANIF=1C:3
 RTD  NAME=VPR23B04 DEVNO=3B04 CHANIF=1G:0
 RTD  NAME=VPR23B05 DEVNO=3B05 CHANIF=1G:1
 RTD  NAME=VPR23B06 DEVNO=3B06 CHANIF=1G:2
 RTD  NAME=VPR23B07 DEVNO=3B07 CHANIF=1G:3
 RTD  NAME=VPR23B08 DEVNO=3B08 CHANIF=1K:0
 RTD  NAME=VPR23B09 DEVNO=3B09 CHANIF=1K:1
 RTD  NAME=VPR23B0A DEVNO=3B0A CHANIF=1K:2
 RTD  NAME=VPR23B0B DEVNO=3B0B CHANIF=1K:3
VTD LOW=9900 HIGH=99FF
 CLUSTER NAME=CLUSTER1 VTSSs(VSMPR1,VSMPR2)
 CLINK VTSS=VSMPR1 CHANIF=0O:0
CLINK VTSS=VSMPR1 CHANIF=0O:1
 CLINK VTSS=VSMPR2 CHANIF=1O:0
CLINK VTSS=VSMPR2 CHANIF=1O:1
```

FIGURE A-13 CONFIG example: Dual ACS Bi-Directional Clustered VTSS System (Part 2)

3. **Specify the Conditional Replication setting on the CONFIG GLOBAL statement.**

```
CONFIG GLOBAL REPLICAT=CHANGED
```

FIGURE A-14 CONFIG GLOBAL Setting for VTV Replication

As with the uni-directional example, in FIGURE A-14, we use CONFIG GLOBAL REPLICAT=CHANGED.

4. **Create a Management Class that specifies VTV replication and two Storage Class to migrate (duplexed) the replicated VTVs.**

```
MGMT NAME(VSMREPL) REPLICAT(YES) MIGPOL(REPLSTR1,REPLSTR2)
```

FIGURE A-15 Management Class for VTV Replication

FIGURE A-15 should look familiar...replicate VTVs only if changed and not in the other VTSS in the Cluster, migrate duplexed to ACSs 01 and 00 by Storage Classes you will create in Step 5.

5. **Create the Storage Classes for the MVCs that contain the replicated, migrated VTVs.**

```
STOR NAME(REPLSTR1) ACS(01) MEDIA(STK1R) MIRATE(EITHER)
STOR NAME(REPLSTR2) ACS(00) MEDIA(STK1R) MIGRATE(EITHER)
```

FIGURE A-16 Storage Classes for Replicated, Migrated VTVs

In FIGURE A-16, the STORclas statement defines Storage Classes REPLSTR1 and REPLSTR2 referenced in the MIGPOL parameter in Step 4. **Also note** that, to optimize VTSS and RTD resources, the MIGRATE parameters on the Storage Classes allow migrates to come from either VTSS. This is a typical strategy for Bi-Directional, or Peer to Peer VTSS Clusters.

6. **Load the MGMTclas and STORclas control statements with the MGMTDEF command.**

```
MGMTDEF DSN(hsc.parms)
```

FIGURE A-17 MGMTDEF Command to Load Statements

7. **Create a TAPEREQ statement to route the critical data to VSM and assign Management Class VSMREPL to the data.**

```
TAPEREQ DSN(*.PAYROLL.**) MEDIA(VIRTUAL) MGMT(VSMREPL)
```

FIGURE A-18 TAPEREQ Statement to Route Critical Data, Assign Management Class VSMREPL

In FIGURE A-18, the TAPEREQ statement specifies:

- Route data sets with HLQ mask *.PAYROLL.** to VSM...
- .....and assign Management Class VSMREPL that you enabled in Step 4.

Caution – To replicate VTVs, **both VSMPR1** and VSMPR2 must be varied online to VTCS so that VTCS can send control commands to both VTSSs. See on page 80 for more information.

Note – Also note the following:

- You can also use esoteric substitution via SMC TAPEREQ statement or ELS User Exits to route replication jobs to VSM. If an esoteric is substituted that spans all VTDs in **all** Peer VTSSs, then VTCS can continue to correctly influence allocation if a one of the Peer VTSSs in a Cluster is taken offline.
- For SMC, a Management Class name, if it is assigned in the StorageTek DFSMS Interface, is available at allocation time. Therefore the esoteric assigned in the interface no longer needs to contain only VTSSs that are part of clusters. As long as the esoteric contains some drives located on the Primary of a full function cluster, SMC has sufficient information to direct allocation to a drive on a Primary VTSS if the Management Class specifies replication enabled.

8. **Check your HSC PARMLIB options to ensure that subtype 28 records are enabled.**

   If enabled, VTSS clustering writes a subtype 28 record for each replication performed.

...and chalk up another success story using Clustered VTSS.

Extended Clustering

Extended Clustering (EC) allows three or more VTSSs to be connected by CLINKs within a single Tapeplex (1 CDS) configuration as shown in the example in FIGURE A-19

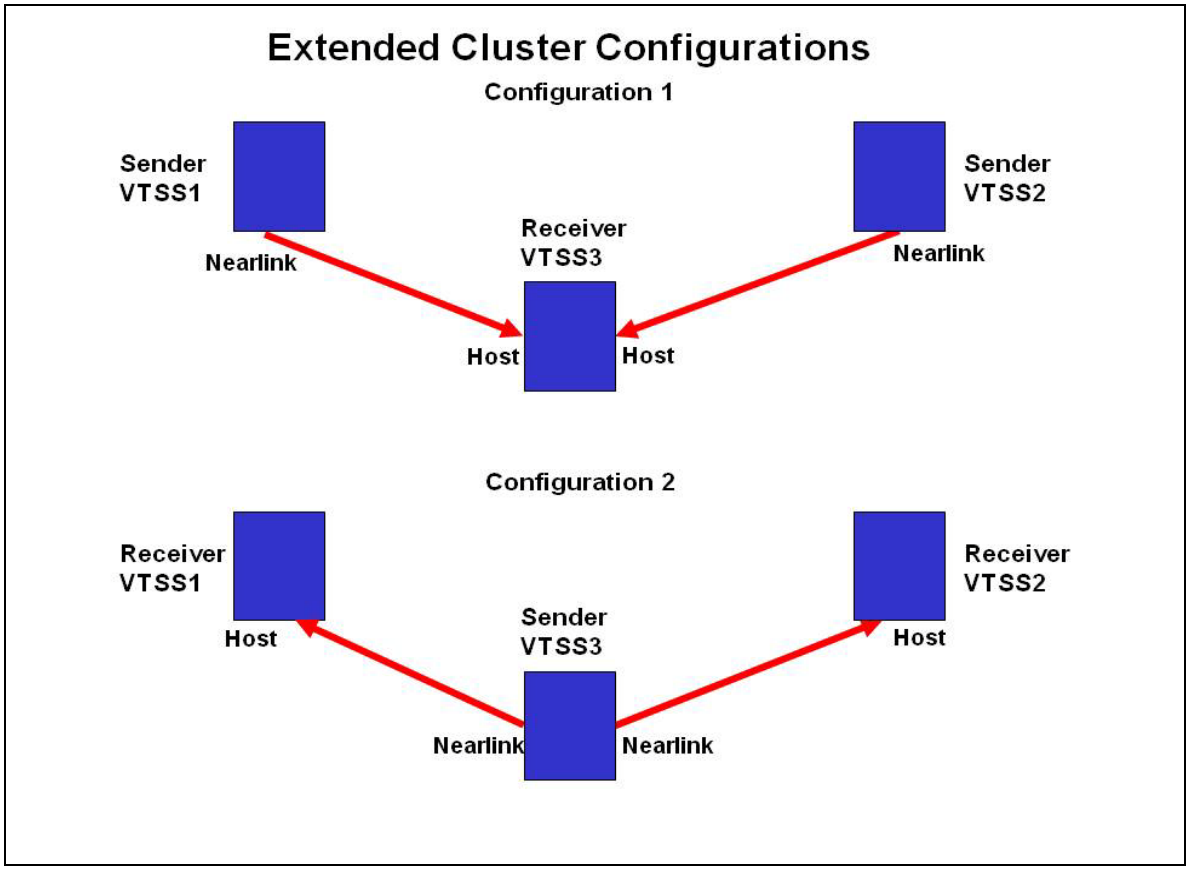

FIGURE A-19 Basic Extended Cluster Configurations

Configuring and Managing a 3 VTSS Clustered System

As shown in FIGURE A-19 on page 172, Configuration 1 shows 2 VTSSs replicating to a single “Collector” VTSS which is the most practical configuration because a primary location with multiple VSMs can feed VTVs to a secondary location with a single collector VSM. Both Synchronous and Asynchronous replication are available to be used on each Sender VTSS. Each VTSS must have equivalent (like model) RTDs connected. As shown in the CONFIG statements for Configuration 1 in FIGURE A-20 on page 174:

- The Cluster statement defines all the VTSS names configured for clustering.
- The Clink statement defines the Nearlink port location on the Sending VTSS and its PARTNER or destination VTSS.

```
/CREATCFG EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL DD DSN=hlq.DBASEPRM,DISP=SHR
//SLSCNTL2 DD DSN=hlq.DBASESEC,DISP=SHR
//SLSSTBY DD DSN=hlq.DBASETBY,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSIN DD *
CONFIG RESET CDSLEVEL(V62ABOVE)
GLOBAL MAXVTV=65000 MVCFREE=60 VTVATTR=SCRATCH RECALWER=YES
LOCKSTR=STK_VTCS_LOCKS VTVPAGE=LARGE
RECLAIM THRESHLD=70 MAXMVC=30 START=40 CONMVC=5
VTSS NAME=VTSS1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD NAME=PA11A00 DEVNO=1A00 CHANIF=0C
RTD NAME=PA11A01 DEVNO=1A01 CHANIF=0D
RTD NAME=PA11A02 DEVNO=1A02 CHANIF=0K
RTD NAME=PA11A03 DEVNO=1A03 CHANIF=0L
RTD NAME=PA12A08 DEVNO=2A08 CHANIF=1C
RTD NAME=PA12A09 DEVNO=2A09 CHANIF=1D
RTD NAME=PA12A0A DEVNO=2A0A CHANIF=1K
RTD NAME=PA12A0B DEVNO=2A0B CHANIF=1L
VTD LOW=7900 HIGH=79FF
VTSS NAME=VTSS2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD NAME=PA23A00 DEVNO=3A00 CHANIF=0C
RTD NAME=PA23A01 DEVNO=3A01 CHANIF=0D
RTD NAME=PA23A02 DEVNO=3A02 CHANIF=0K
RTD NAME=PA23A03 DEVNO=3A03 CHANIF=0L
RTD NAME=PA24A08 DEVNO=4A08 CHANIF=1C
RTD NAME=PA24A09 DEVNO=4A09 CHANIF=1D
RTD NAME=PA24A0A DEVNO=4A0A CHANIF=1K
RTD NAME=PA24A0B DEVNO=4A0B CHANIF=1L
VTD LOW=8900 HIGH=89FF
VTSS NAME=VTSS3 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD NAME=PA33A00 DEVNO=3A00 CHANIF=0C
RTD NAME=PA33A01 DEVNO=3A01 CHANIF=0D
RTD NAME=PA33A02 DEVNO=3A02 CHANIF=0K
RTD NAME=PA33A03 DEVNO=3A03 CHANIF=0L
RTD NAME=PA34A08 DEVNO=4A08 CHANIF=1C
RTD NAME=PA34A09 DEVNO=4A09 CHANIF=1D
RTD NAME=PA34A0A DEVNO=4A0A CHANIF=1K
RTD NAME=PA34A0B DEVNO=4A0B CHANIF=1L
VTD LOW=9900 HIGH=99FF
CLUSTER NAME=CLUSTER1 VTSS(VTSS1,VTSS2,VTSS3)
CLINK VTSS=VTSS1 CHANIF=0G PART=VTSS3
CLINK VTSS=VTSS2 CHANIF=0G PART=VTSS3
```

FIGURE A-20 CONFIG Statements for Extended Cluster Configuration 1

As shown in FIGURE A-19 on page 172, Configuration 2 shows a Single replicating VTSS connected to 2 receiver VTSSs. Note that the term “Collector” was not used here because a VTV is only replicated to one VTSS either VTSS1 or VTSS2 but not both and the receiver VTSS is not configurable. This is a very important concept to under stand because there are not currently any Management Class parameters that will select a specific VTSS to direct a VTV. This configuration is not useful for implementation in a Primary and Secondary Location environment where the VTV must end up at a specific secondary location and may make Extended Bidirectional configurations undesirable. Both Synchronous and Asynchronous replication are available to be used on the Sender VTSS. Each VTSS must have equivalent (like model) RTDs connected.

Configuration 2 becomes most important when deciding to implement Bi-directional replication in an Extended Cluster environment. Bi-directional replication is required then use the “many VTSSs to one VTSS” configuration in one direction and “VTSS Pair” configuration in the other direction where the “VTSS Pair” is configure between the two VTSSs the replicated VTV must reside on.

As shown in the CONFIG statements for Configuration 2 in FIGURE A-21 on page 176:

- The Cluster statement defines all the VTSS names configured for clustering.
- The Clink statement defines the Nearlink port location on the Sending VTSS and its PARTNER or destination VTSS.

```
//CREATCFG EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL DD DSN=hlq.DBASEPRM,DISP=SHR
//SLSCNTL2 DD DSN=hlq.DBASESEC,DISP=SHR
//SLSSTBY DD DSN=hlq.DBASETBY,DISP=SHR
//SLSPRINT DD SYSOUT=*
//SLSIN DD *
CONFIG RESET CDSLEVEL(V62ABOVE)
GLOBAL MAXVTV=65000 MVCFREE=60 VTVATTR=SCRATCH RECALWER=YES
LOCKSTR=STK_VTCS_LOCKS VTVPAGE=LARGE
RECLAIM THRESHLD=70 MAXMVC=30 START=40 CONMVC=5
VTSS NAME=VTSS1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD NAME=PA11A00 DEVNO=1A00 CHANIF=0C
RTD NAME=PA11A01 DEVNO=1A01 CHANIF=0D
RTD NAME=PA11A02 DEVNO=1A02 CHANIF=0K
RTD NAME=PA11A03 DEVNO=1A03 CHANIF=0L
RTD NAME=PA12A08 DEVNO=2A08 CHANIF=1C
RTD NAME=PA12A09 DEVNO=2A09 CHANIF=1D
RTD NAME=PA12A0A DEVNO=2A0A CHANIF=1K
RTD NAME=PA12A0B DEVNO=2A0B CHANIF=1L
VTD LOW=7900 HIGH=79FF
VTSS NAME=VTSS2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD NAME=PA23A00 DEVNO=3A00 CHANIF=0C
RTD NAME=PA23A01 DEVNO=3A01 CHANIF=0D
RTD NAME=PA23A02 DEVNO=3A02 CHANIF=0K
RTD NAME=PA23A03 DEVNO=3A03 CHANIF=0L
RTD NAME=PA24A08 DEVNO=4A08 CHANIF=1C
RTD NAME=PA24A09 DEVNO=4A09 CHANIF=1D
RTD NAME=PA24A0A DEVNO=4A0A CHANIF=1K
RTD NAME=PA24A0B DEVNO=4A0B CHANIF=1L
VTD LOW=8900 HIGH=89FF
VTSS NAME=VTSS3 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTD NAME=PA33A00 DEVNO=3A00 CHANIF=0C
RTD NAME=PA33A01 DEVNO=3A01 CHANIF=0D
RTD NAME=PA33A02 DEVNO=3A02 CHANIF=0K
RTD NAME=PA33A03 DEVNO=3A03 CHANIF=0L
RTD NAME=PA34A08 DEVNO=4A08 CHANIF=1C
RTD NAME=PA34A09 DEVNO=4A09 CHANIF=1D
RTD NAME=PA34A0A DEVNO=4A0A CHANIF=1K
RTD NAME=PA34A0B DEVNO=4A0B CHANIF=1L
VTD LOW=9900 HIGH=99FF
CLUSTER NAME=CLUSTER1 VTSS(VTSS1,VTSS2,VTSS3)
CLINK VTSS=VTSS3 CHANIF=0G PART=VTSS1
CLINK VTSS=VTSS3 CHANIF=0G PART=VTSS2
```

FIGURE A-21 CONFIG Statements for Extended Cluster Configuration 2

VSM5 to VSM5 Cluster with TCP/IP CLINKs

FIGURE A-22 shows an example of a VSM5 to VSM5 Cluster with TCP/IP CLINKs.

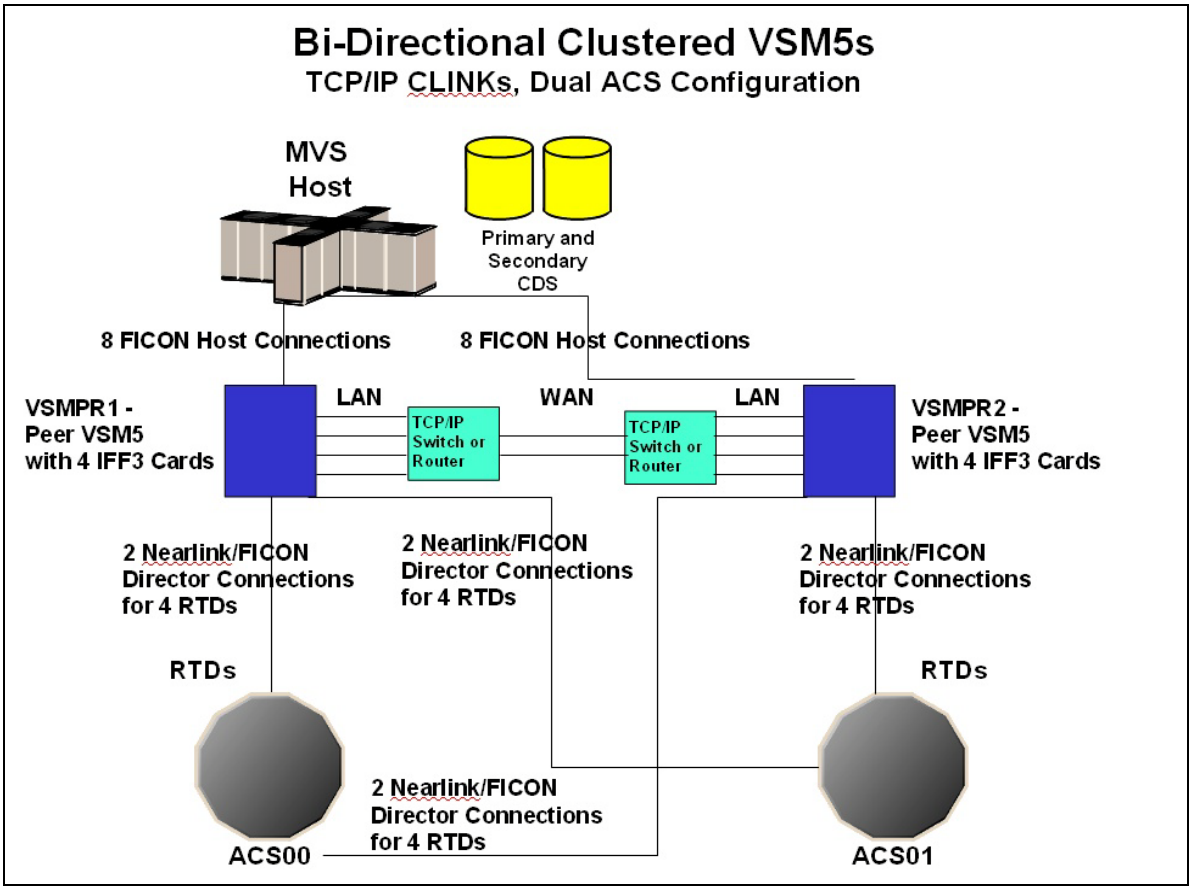

FIGURE A-22 Clustered VSM5s with TCP/IP CLINKs

In FIGURE A-22 on page 177, assume that for redundancy, you will use targets on separate IFF cards on each VSM5 for Native IP as shown in TABLE A-1 and TABLE A-2.

TABLE A-1 CLINK IPIF Values for VSMPR1

IFF Card	Target Number	Example IP	Corresponding CLINK IPIF
IFF0	**Target 0**	128.0.1.1	0A:0
IFF1	**Target 0**	128.0.2.1	0I:0
IFF2	**Target 0**	128.0.3.1	1A:0
IFF3	**Target 0**	128.0.4.1	1I:0

TABLE A-2 CLINK IPIF Values for VSMPR2

IFF Card	Target Number	Example IP	Corresponding CLINK IPIF
IFF0	**Target 0**	128.0.1.2	0A:0
IFF1	**Target 0**	128.0.2.2	0I:0
IFF2	**Target 0**	128.0.3.2	1A:0
IFF3	**Target 0**	128.0.4.2	1I:0

FIGURE A-23 on page 179 shows example CONFIG JCL to define the configuration shown in FIGURE A-22 on page 177 with the values shown in TABLE A-1 and TABLE A-2.

```
//CREATECF EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB      DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL      DD DSN=FEDB.VSMLMULT.DBASEPRM,DISP=SHR
//SLSCNTL2     DD DSN=FEDB.VSMLMULT.DBASESEC,DISP=SHR
//SLSSTBY      DD DSN=FEDB.VSMLMULT.DBASETBY,DISP=SHR
//SLSPRINT     DD   SYSOUT=*
//SLSIN        DD   *
 CONFIG CDSLEVEL(V61ABOVE)
GLOBAL MAXVTV=32000 MVCFREE=40 VTVATTR=SCRATCH RECALWER=YES
LOCKSTR=VTCS_LOCKS REPLICAT=ALWAYS VTVPAGE=LARGE INITMVC=YES
SYNCHREP=YES MAXRTDS=16 FASTMIGR=YES
RECLAIM      THRESHLD=70 MAXMVC=40  START=35
VTSS NAME=VSMPR1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
VTD LOW=8900 HIGH=89FF
RTD  NAME=VPR12A00 DEVNO=2A00 CHANIF=0C:0
 RTD  NAME=VPR12A01 DEVNO=2A01 CHANIF=0C:1
 RTD  NAME=VPR12A02 DEVNO=2A02 CHANIF=0C:2
 RTD  NAME=VPR12A03 DEVNO=2A03 CHANIF=0C:3
 RTD  NAME=VPR12A04 DEVNO=2A04 CHANIF=0G:0
 RTD  NAME=VPR12A05 DEVNO=2A05 CHANIF=0G:1
 RTD  NAME=VPR12A06 DEVNO=2A06 CHANIF=0G:2
 RTD  NAME=VPR12A07 DEVNO=2A07 CHANIF=0G:3
VTSS NAME=VSMPR2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
VTD LOW=9900 HIGH=99FF
RTD  NAME=VPR22B00 DEVNO=2B00 CHANIF=0C:0
 RTD  NAME=VPR22B01 DEVNO=2B01 CHANIF=0C:1
 RTD  NAME=VPR22B02 DEVNO=2B02 CHANIF=0C:2
 RTD  NAME=VPR22B03 DEVNO=2B03 CHANIF=0C:3
 RTD  NAME=VPR22B04 DEVNO=2B04 CHANIF=0G:0
 RTD  NAME=VPR22B05 DEVNO=2B05 CHANIF=0G:1
 RTD  NAME=VPR22B06 DEVNO=2B06 CHANIF=0G:2
 RTD  NAME=VPR22B07 DEVNO=2B07 CHANIF=0G:3
CLUSTER NAME=CLUSTER1 VTSSs(VSMPR1,VSMPR2)
 CLINK VTSS=VSMPR1 IPIF=0A:0
 CLINK VTSS=VSMPR1 IPIF=0I:0
CLINK VTSS=VSMPR1 IPIF=1A:0
 CLINK VTSS=VSMPR1 IPIF=1I:0
CLINK VTSS=VSMPR2 IPIF=0A:0
 CLINK VTSS=VSMPR2 IPIF=0I:0
CLINK VTSS=VSMPR2 IPIF=1A:0
 CLINK VTSS=VSMPR2 IPIF=1I:0
```

FIGURE A-23 CONFIG example: Clustered VSM5s with TCP/IP CLINKs

VSM5 to VSM 6 Cluster with TCP/IP CLINKs, Cross-Connected VLEs

FIGURE A-24 shows an example of a VSM5 to VSM 6 Cluster with TCP/IP CLINKs, where each VTSS is cross-connected to two VLEs.

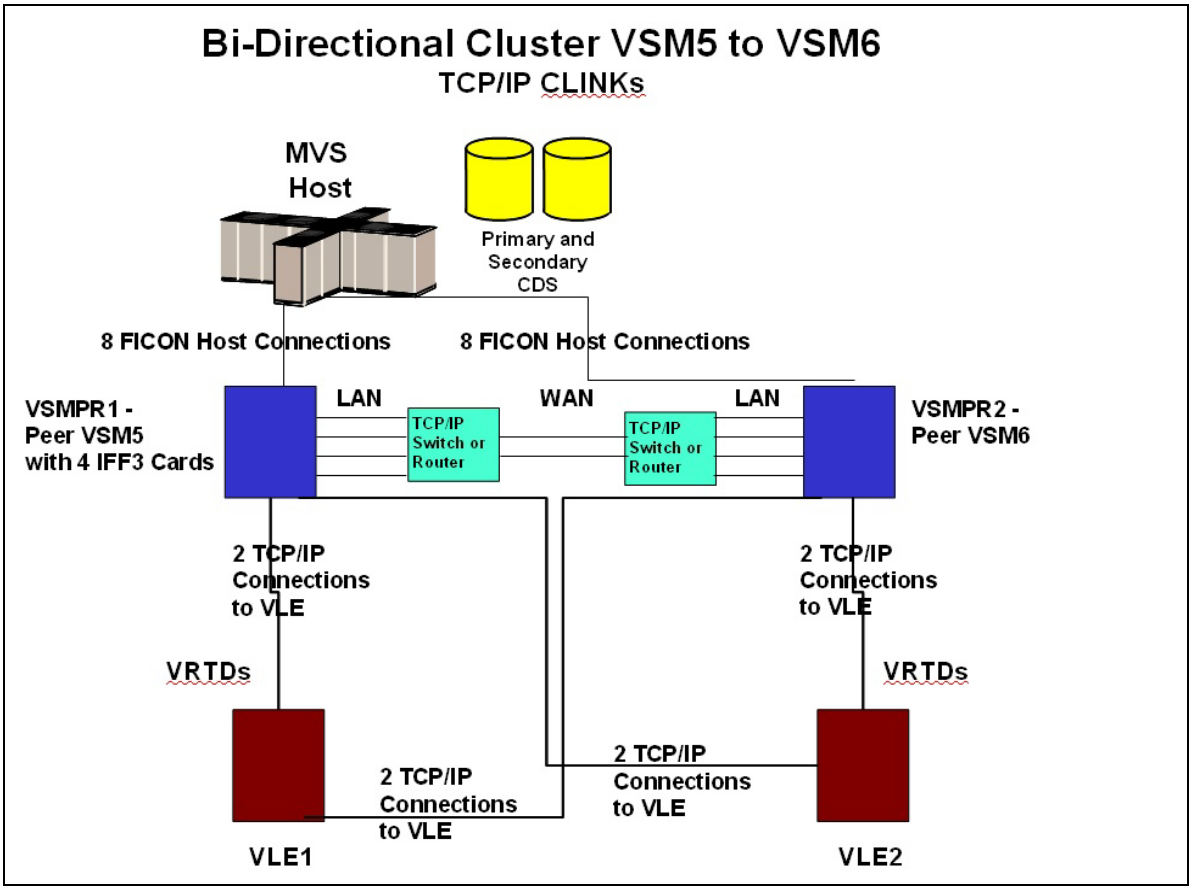

FIGURE A-24 VSM5 to VSM 6 Cluster with TCP/IP CLINKs, Cross-Connected VLEs

In FIGURE A-24 on page 180, assume that for redundancy, you will use targets on separate IFF cards for the VSM5 (VSMPR1) for Native IP and for VLE connections as shown in TABLE A-3 andTABLE A-4.

TABLE A-3 CLINK IPIF Values for VSMPR1

IFF Card	Target Number	Example IP	Corresponding CLINK IPIF
IFF0	**Target 0**	128.0.1.1	0A:0
IFF1	**Target 0**	128.0.2.1	0I:0
IFF2	**Target 0**	128.0.3.1	1A:0
IFF3	**Target 0**	128.0.4.1	1I:0

TABLE A-4 RTD IPIF Values for VSMPR1

IFF Card	Target Number	Example IP	Corresponding CLINK IPIF
IFF0	**Target 1**	128.0.1.2	0A:1
IFF1	**Target 1**	128.0.2.2	0I:1
IFF2	**Target 1**	128.0.3.2	1A:1
IFF3	**Target 1**	128.0.4.2	1I:1

FIGURE A-25 on page 182 shows example CONFIG JCL to define the configuration shown in FIGURE A-24 on page 180 with the values shown in TABLE A-3 and TABLE A-4.

```
//CREATECF EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB      DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL      DD DSN=FEDB.VSMLMULT.DBASEPRM,DISP=SHR
//SLSCNTL2     DD DSN=FEDB.VSMLMULT.DBASESEC,DISP=SHR
//SLSSTBY      DD DSN=FEDB.VSMLMULT.DBASETBY,DISP=SHR
//SLSPRINT     DD   SYSOUT=*
//SLSIN        DD   *
 CONFIG CDSLEVEL(V61ABOVE)
GLOBAL MAXVTV=32000 MVCFREE=40 VTVATTR=SCRATCH RECALWER=YES
LOCKSTR=VTCS_LOCKS REPLICAT=ALWAYS VTVPAGE=LARGE INITMVC=YES
SYNCHREP=YES MAXRTDS=16 FASTMIGR=YES
RECLAIM      THRESHLD=70 MAXMVC=40  START=35
VTSS NAME=VSMPR1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
VTD LOW=8900 HIGH=89FF
RTD  NAME=VL1RTD1 STORMNGR=VLE1 IPIF=0A:1
RTD  NAME=VL1RTD2 STORMNGR=VLE1 IPIF=0I:1
RTD  NAME=VL2RTD1 STORMNGR=VLE2 IPIF=1A:1
RTD  NAME=VL2RTD2 STORMNGR=VLE2 IPIF=1I:1
VTSS NAME=VSMPR2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
VTD LOW=9900 HIGH=99FF
RTD  NAME=VL1RTD1 STORMNGR=VLE1 IPIF=0A:1
RTD  NAME=VL1RTD2 STORMNGR=VLE1 IPIF=0I:1
RTD  NAME=VL2RTD1 STORMNGR=VLE2 IPIF=1A:1
RTD  NAME=VL2RTD2 STORMNGR=VLE2 IPIF=1I:1
CLUSTER NAME=CLUSTER1 VTSSs(VSMPR1,VSMPR2)
 CLINK VTSS=VSMPR1 IPIF=0A:0
 CLINK VTSS=VSMPR1 IPIF=0I:0
CLINK VTSS=VSMPR1 IPIF=1A:0
 CLINK VTSS=VSMPR1 IPIF=1I:0
CLINK VTSS=VSMPR2 IPIF=0A:0
 CLINK VTSS=VSMPR2 IPIF=0I:0
CLINK VTSS=VSMPR2 IPIF=1A:0
 CLINK VTSS=VSMPR2 IPIF=1I:0
```

FIGURE A-25 CONFIG example: VSM5 to VSM 6 Cluster with TCP/IP CLINKs

In FIGURE A-25, note that while the CLINK IPIF and RTD IPIF parameter values for the VSM5 (VSMPR1) must match the values shown in TABLE A-3 on page 181 and TABLE A-4 on page 181, the CLINK IPIF and RTD IPIF values for the VSM 6 (VSMPR2) only have to meet VTCS restrictions on these values and must be unique for each VTSS; they **do not** correspond to an actual value on the VSM 6 TCP/IP ports!

VSM 6 to VSM 6 “Tapeless” Cluster with TCP/IP CLINKs

FIGURE A-26 shows an example of a “Tapeless” VSM 6 to VSM 6 Cluster with TCP/IP CLINKs.

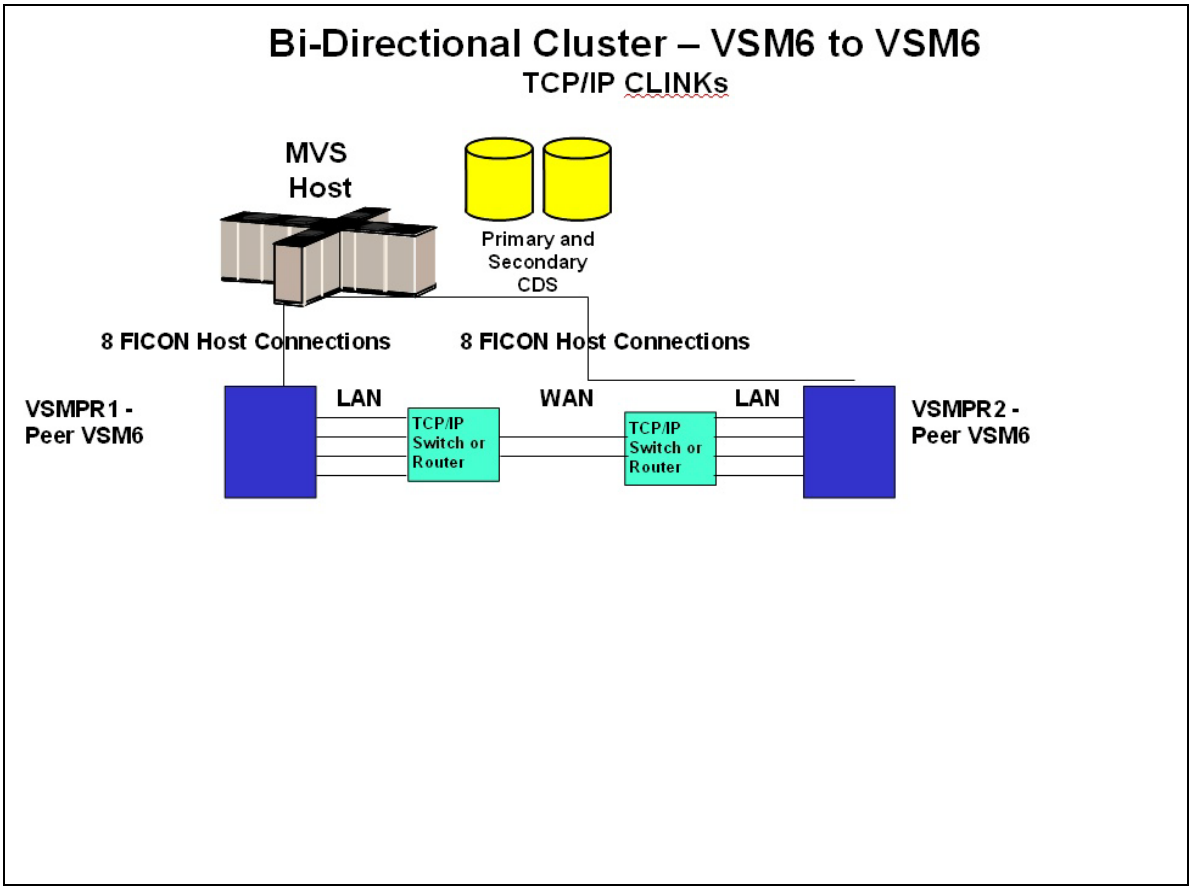

FIGURE A-26 VSM 6 to VSM 6 Tapeless Cluster with TCP/IP CLINKs

FIGURE A-27 on page 184 shows example CONFIG JCL to define the configuration shown in FIGURE A-26.

```
//CREATECF EXEC PGM=SLUADMIN,PARM='MIXED'
//STEPLIB      DD DSN=hlq.SEALINK,DISP=SHR
//SLSCNTL      DD DSN=FEDB.VSMLMULT.DBASEPRM,DISP=SHR
//SLSCNTL2     DD DSN=FEDB.VSMLMULT.DBASESEC,DISP=SHR
//SLSSTBY      DD DSN=FEDB.VSMLMULT.DBASETBY,DISP=SHR
//SLSPRINT     DD   SYSOUT=*
//SLSIN        DD   *
 CONFIG CDSLEVEL(V61ABOVE)
GLOBAL MAXVTV=32000 MVCFREE=40 VTVATTR=SCRATCH RECALWER=YES
LOCKSTR=VTCS_LOCKS REPLICAT=ALWAYS VTVPAGE=LARGE INITMVC=YES
SYNCHREP=YES MAXRTDS=16 FASTMIGR=YES
RECLAIM      THRESHLD=70 MAXMVC=40  START=35
VTSS NAME=VSMPR1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
VTD LOW=8900 HIGH=89FF
VTSS NAME=VSMPR2 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
VTD LOW=9900 HIGH=99FF
CLUSTER NAME=CLUSTER1 VTSSs(VSMPR1,VSMPR2)
 CLINK VTSS=VSMPR1 IPIF=0A:0
 CLINK VTSS=VSMPR1 IPIF=0A:1
CLINK VTSS=VSMPR1 IPIF=0A:2
 CLINK VTSS=VSMPR1 IPIF=0A:3
CLINK VTSS=VSMPR2 IPIF=0A:0
 CLINK VTSS=VSMPR2 IPIF=0A:1
CLINK VTSS=VSMPR2 IPIF=0A:2
 CLINK VTSS=VSMPR2 IPIF=0A:3
```

FIGURE A-27 CONFIG example: VSM 6 to VSM 6 Cluster with TCP/IP CLINKs

In FIGURE A-27, note that the CLINK IPIF values for both VSM 6s only have to meet VTCS restrictions on these values and must be unique for each VTSS; they **do not** correspond to an actual value on the VSM 6 TCP/IP ports! **Also note that** because the cluster is tapeless, there are no CONFIG RTD statements for either VSM 6.

Variation on a Theme: Uni-Directional or Bi-Directional?...

...the choice is yours! This is yet another variation on the theme we showed with Bi-Directional Clustering. We're going to use VTSSLST and VTSSSEL statements, however, to make a Bi-Directional Cluster Uni-Directional. Why would I want to do this? What if I wanted to switch the roles of the Primary and Secondary VTSSs? Easy, you just start with the same setup as described in the procedure beginning on "Configuring and Managing a Bi-Directional Clustered System" on page 167. After you complete Step 5, you throw in a subtle change with the following VTSSLST and VTSSSEL statements.

```
VTSSLST NAME(SITEA) VTSS(VSMPR1)
VTSSSEL FUNCTION(SCRATCH) HOST(MVSA) VTSSLST(SITEA)
VTSSSEL FUNCTION(SPECIFIC) HOST(MVSA) VTSSLST(SITEA)
```

FIGURE A-28 VTSSLST/VTSSSEL Statements - VSMPR1 Primary, VSMPR2 Secondary

In FIGURE A-28:

- The VTSSLST statement defines VTSS list SITEA that contains **only** VSMPR1.
- The VTSSSEL statements direct scratch and specific VTV mounts from MVSA to SITEA, which contains **only** VSMPR1...thus effectively making it the Primary.

So this Cluster is actually Bi-Directional, but VTSSLST and VTSSSEL statements give us the flexibility to effectively make either VTSS the Primary and the other the Secondary by simply loading the corresponding MGMTclas, STORclas, VTSSLST, and VTSSSEL control statements with the MGMTDEF command.

What if we wanted to switch the Primary and Secondary? No problem, just rewrite the VTSSLST and VTSSSEL statements to make VSMPR2 the Primary and VSMPR1 the Secondary.

```
VTSSLST NAME(SITEB) VTSS(VSMPR2)
VTSSSEL FUNCTION(SCRATCH) HOST(MVSB) VTSSLST(SITEB)
VTSSSEL FUNCTION(SPECIFIC) HOST(MVSB) VTSSLST(SITEB)
```

FIGURE A-29 VTSSLST/VTSSSEL Statements - VSMPR2 Primary, VSMPR1 Secondary

In FIGURE A-29:

- The VTSSLST statement defines list SITEB that contains **only** VSMPR2.
- The VTSSSEL statements direct scratch and specific VTV mounts from MVSB to SITEB, which contains **only** VSMPR2...thus effectively making it the Primary.

Finally, what if the time came that things worked better with this Cluster as a true Bi-Directional Cluster? Easy...just delete the VTSSLST and VTSSSEL statements, reload your defs, and you're all set...

Now *that's* flexibility!