Virtual Library Extension

Configuring the MVS Host Software for VLE

Version 1.3

ORACLE

Configuring the MVS Host Software for VLE

E39445-04

Table of Contents

Preface

Audience

This guide is for StorageTek or customer personnel who are responsible for configuring theMVS host software for Oracle's StorageTek™ Virtual Library Extension (VLE).

Audience

1

What is Virtual Library Extension?

Oracle's StorageTek™ Virtual Library Extension (VLE) is back-end disk storage for VTSS. VLE provides:

- An additional storage tier in the VSM solution. VTVs can now migrate from VTSS to VLE to provide fast access to recent data. Additionally, VTVs can transition from VLE storage to tape media (MVCs) for long term archive.You can control how VTVs are migrated and archived via the existing HSC Management and Storage Classes, providing full backward compatibility with previous configurations.

- Back-end disk storage shared between multiple VTSS systems ensuring high-availability access to data.

  Note – For VLE 1.1 and above, a "VLE" is a collection of nodes interconnected with a private network.

To VTCS, a VLE looks like a tape library except that the VTVs are stored in Virtual Multi-Volume Cartridges (VMVCs) on disk. With VLE, you can configure either a VLE and tape or a VLE only (for example, with Tapeless VSM configurations) back-end VTV storage solution. A VTSS can migrate VTVs to and recall them from a VLE, just as is done with a real tape library.

Caution –

- **Note that** if you have a VLE system, HSC/VTCS uses SMC communication services to communicate with the VLE. To ensure that these services are available during VTCS startup, Oracle recommends that you first issue the start command for HSC, then immediately issue the start command for SMC, while HSC is initializing.

- **Also note that** stopping SMC stops VTCS from sending messages to the VLE, which effectively stops data transfer. Therefore, you should ensure that VTCS activity is quiesced or VTCS is terminated before stopping SMC.

- You cannot use AT-TLS with the SMC HTTP server if you are using VLE.

- **Note that** in Tapeless VSM configurations, if you have only a single-node VLE attached to a specific VTSS and that VLE goes offline, you lose access to any VTVs migrated to the VLE that are not resident in the VTSS until the VLE comes back online.

The VLE solution consists of:

- Virtual Tape Storage Subsystem (VTSS) hardware and microcode.
- Virtual Tape Control Subsystem (VTCS) software and Storage Management Component (SMC).
- VLE hardware and software.

Single Node VLE Configuration

FIGURE 1-1 shows a single node VLE configuration.
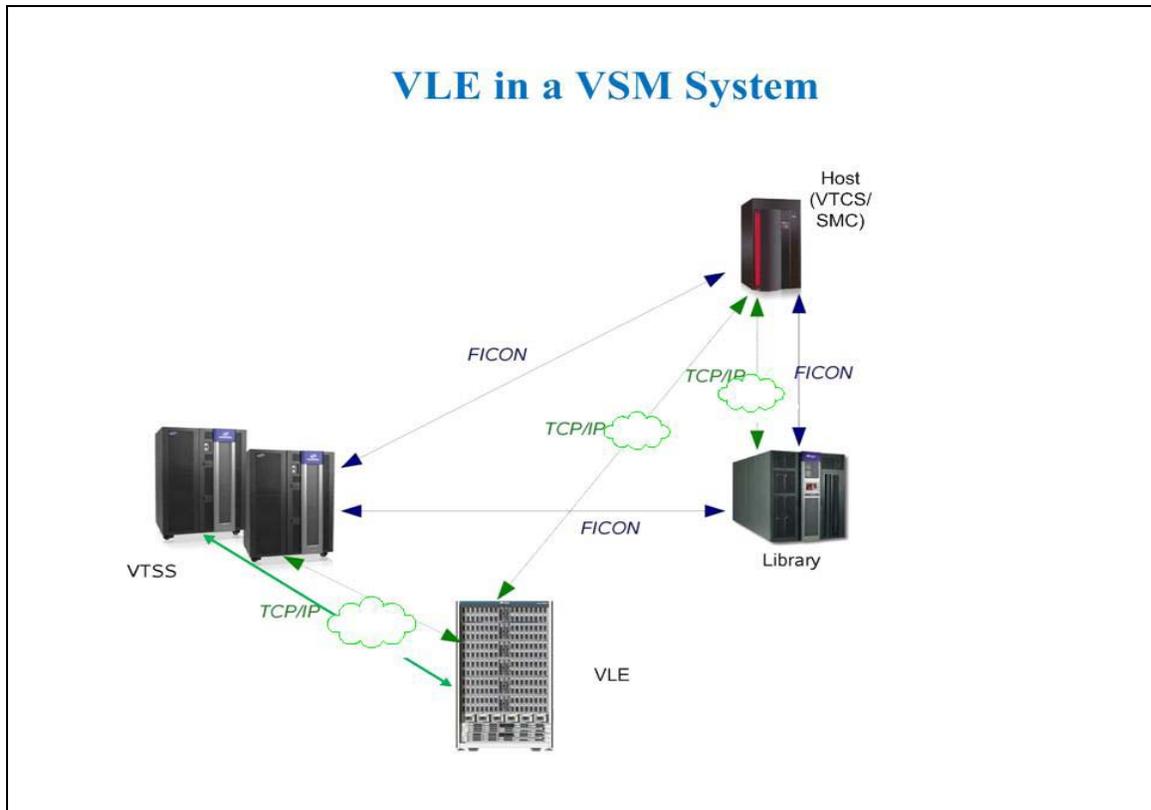
FIGURE 1-1 Single Node VLE in a VSM System

As FIGURE 1-1 shows:

- Multiple TCP/IP connections (between the VTSS's IP ports and the VLE's IP ports) are supported as follows:

    - A single VLE can connect up to 8 VTSSs, so VTSSs can share VLEs.

    - A single VTSS can connect to up to 4 VLEs to increase buffer space for heavy workloads.

- A single VTSS can be attached to:

    - Only RTDs

    - Only other VTSSs (clustered)

    - Only VLEs

    - Any combination of the above.

- TCP/IP is the only supported protocol for connections between the VLE and the VTSS and for connections between the VLE and hosts running SMC and VTCS.

Multi-Node VLE Systems

- Multi-node VLE systems **enable massive scaling of the VLE storage system**. You can construct *multi-node systems* that can consist of one to 64 nodes, with multiple nodes interconnected by a private network. A multi-node VLE appears to SMC/VTCS as a single VLE. A single VLE, therefore, can scale between 330 TB (for a single-node VLE with two JBODs) and 82.5 PB (for a fully populated 64-node VLE).

  Note – These are effective capacities, assuming 4:1 compression. **Also note** that VLE is **architected** for up to 64 nodes, but has only been **validated** for up to 7 nodes.

shows a VLE multi-node complex, where the nodes are cross connected into a dedicated 10GE switch so that each node can access any other node in the complex.
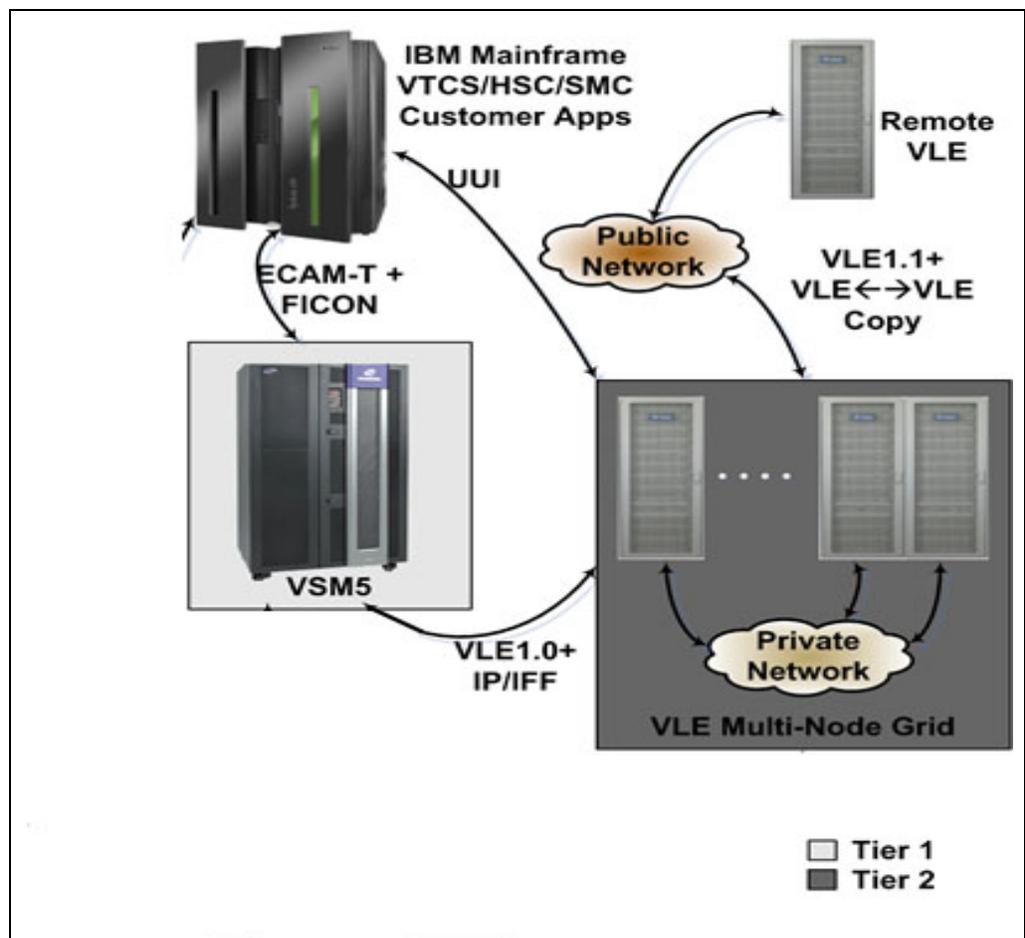
FIGURE 1-2 VLE Multi-Node Complex

VLE to VLE Data Transfer

The VLE storage system can manage data transfers independently of the VTSS, which frees VTSS resources for front-end (host) workload, which improves the overall VTSS through-put. For example:

- If your migration policies specify that there should be two VLE copies of a VTV (either in the same or separate VLEs), then the first migrate to a VLE will cause data to be transferred from the VTSS and all subsequent VLE migrates for the VTV may be achieved via a VLE to a VLE copy. This reduces the VTSS cycle times required to migrate all copies of a VTV.

- If your environment runs:

    - VLE 1.2 or above, and

    - VTCS 7.1 (with the supporting PTFs) or VTCS 7.2

    Then you can use VTCS to define more VLE devices than there are VTSS to VLE paths via the `CONFIG STORMNGR VLEDEV` parameter. If you use this addressing scheme, then the VTSS resources used to migrate all the VTV copies to VLE are reduced even further because the path from the VTSS to the target VLE is only reserved when the data transfer is direct from the VTSS to the VLE. For all VLE VRTD actions, a path from the VTSS is only reserved when VTSS data transfer is required.

VTV Encryption

The encryption feature enables encryption of VMVCs written to the VLE system. Encryption is enabled on a *per node basis*, via an *encryption key* stored on the node, backed up on a USB device. Encryption is entirely managed via the VLE GUI; the host software has no knowledge of encryption, as the VLE deencrypts VTVs that are recalled to the VTSS.

VTV Deduplication

Deduplication eliminates redundant data in a VLE complex. Deduplication, which is controlled by the `STORCLAS` statement `DEDUP` parameter, increases the effective VLE capacity and is performed by the VLE before the VTV is written to a VMVC.

To assess deduplication results, enable deduplication, monitor the results with the `SCRPT` report, and fine tune deduplication as necessary. The `SCRPT` report provides the approximate "reduction ratio" for the deduplicated data, which is uncompressed Gb divided by used Gb. The Reduction Ratio, therefore, includes **both** VTSS compression **and** VLE deduplication. A larger reduction ratio indicates more effective compression and deduplication.

For example, the VTSS receives 16 Mb of data, compresses it to 4Mb, and writes the compressed data to a VTV. VLE subsequently deduplicates the VTV to 2Mb and writes it to a VMVC. Thus, the reduction ratio is 16Mb divided by 2Mb or 8.0:1.

Early Time To First Byte (ETTFB)

Early Time To First Byte (ETTFB), also known as the concurrent tape recall/mount feature, allows the VTSS to use a VTD to read data as it being recalled from VLE:

- ETTFB is set globally via `CONFIG GLOBAL FASTRECL`.

- If `CONFIG GLOBAL FASTRECL=YES`, you can disable ETTFB on per VTSS basis via `CONFIG VTSS NOERLYMNT`.

`CONFIG GLOBAL` and `CONFIG VTSS` apply to **both** ETTFB for RTDs and ETTFB for VLE.

Frame Size Control

Frame Size Control specifies the use of Jumbo frames on each copy link:

- **If** your TCP/IP network supports Jumbo Frames, enabling this option can improve network performance.

- You enable Jumbo Frames via the **Jumbo Frames** check box on the **Port Card Configuration** tab. Selecting this check box sets the MTU (Maximum Transmission Unit) value to 9000 for the port

What's New With VLE 1.3?

VLE 1.3 adds the following features:

- **Reduced replication**, which allows VTVs to be copied between VMVCs in deduplicated format. The only data copied is data that did **not** reside on the destination node when the copy began. Reducing the amount of data copied, therefore, lowers network use and copy times.

  As described in "VTV Deduplication" on page 11, you enable deduplication on a Storage Class basis via the STORCLAS statement DEDUP parameter. Reduced replication, therefore, works as follows:

  - If deduplication is enabled for **both** the source and destination Storage Classes, then VTVs are copied in deduplicated format. This is the **only** configuration required for reduced replication.

  - If deduplication is enabled for the source but **not** the destination Storage Class, then VTVs are "hydrated" (reconstituted) before being copied.

  - If deduplication is enabled for the destination but **not** the source Storage Class, then VTVs are deduplicated when received at the destination.

- The VLE GUI now provides the ability to define **static routes**, which is simpler and less error-prone than the previous method (defining static routes via Solaris commands).

VLE Hardware and Software

The VLE, which is a factory-assembled unit in a Sun Rack II Model 1242, consists of the following hardware:

- A server built on a Sun Server X2-4 platform.
- Four 1GigE ports for a combination of SMC UUI connections and service connections.
- A service (ILOM) port.
- Four Quad-port 1GigE cards, which provide 16 ethernet ports for data transfer.
- J4410 JBODs in a ZFS RAID array, available in effective capacities of 330TB, 660TB, 990TB, or 1320TB (assuming a 4 to 1 compression ratio when the data is migrated to the VLE).
- Two dual-port 10GigE NIC cards per server, which are required for the internal network connections for VLEs with 3 or more nodes.

The VLE software consists of:

- Oracle Solaris 11 Operating System.
- ZFS file system and MySQL database.
- The VLE application software.

FIGURE 1-3 shows the VLE subsystem architecture.
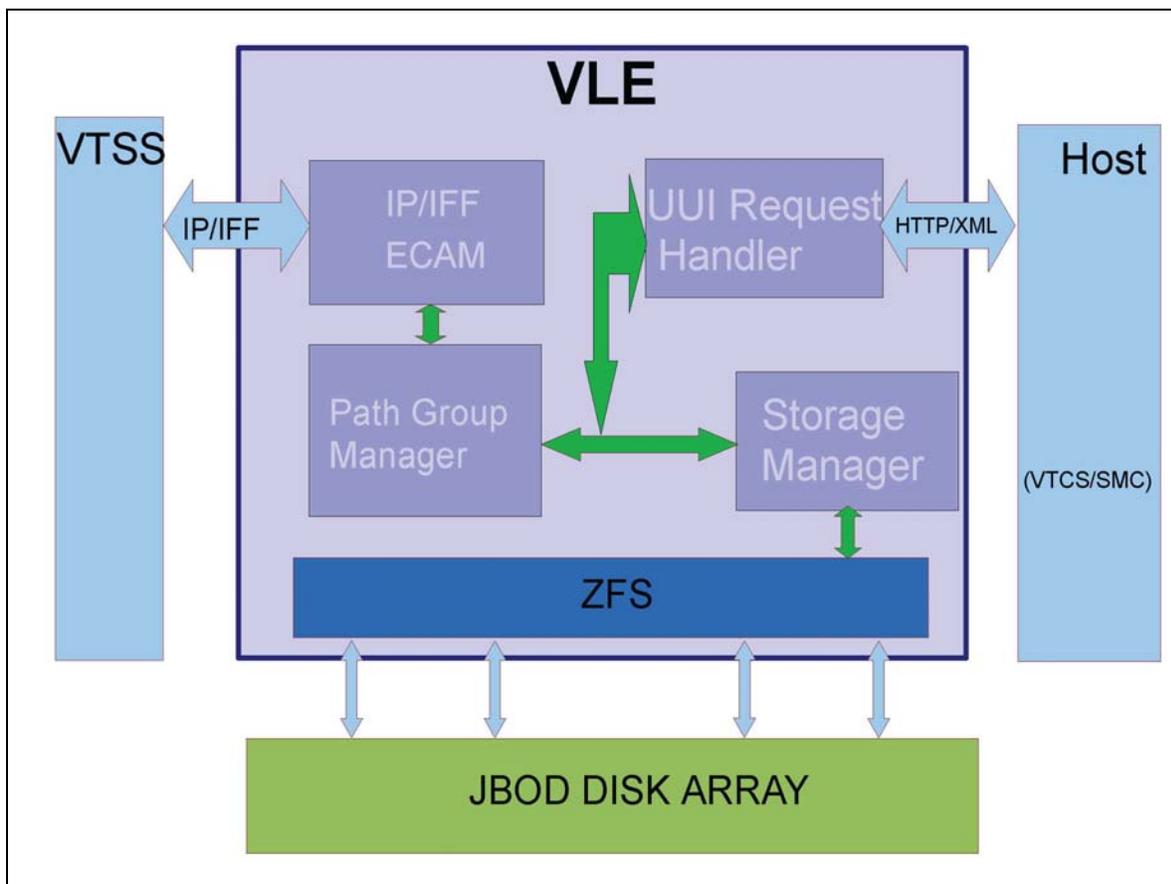
FIGURE 1-3 VLE Subsystem Architecture

As shows, the VLE application software is comprised of:

- HTTP/XML is the data protocol for host to VLE communications.

- The Universal User Interface (UUI) Request Handler, which processes UUI requests from and produces responses to Storage Management Component (SMC) and Virtual Tape Control Software (VTCS). The UUI Request Handler determines which VLE components are used to service a request.

  UUI Request Handler calls:

  - The PathGroup Manager to schedule VTV migrates and recalls. The PathGroup Manager manages all Path Groups, where each Path Group manages a single VTV data transfer between the VTSS and the VLE.

  - The Storage Manager to schedule all report generation.

- The VLE Storage Manager component manages the VMVC/VTV data and meta data on the VLE. The VLE Storage Manager stores VTV data on and retrieves it from the ZFS on the JBOD array.

- TCP/IP/IFF is the data protocol for host to VLE communications, where the IP/IFF/ECAM component handles communications between the VTSS and the VLE.

2

Configuring the MVS Host Software

This chapter provides the MVS host software configuration for VLE as described in the following sections:

- *"Key Configuration Values" on page 18*
- *"MVS Host Software Configuration Tasks" on page 20*

Key Configuration Values

The following sections describes values required for software configuration that must match values that are typically already set in the hardware configuration and recorded in the
`IP_and VMVC_Configuration.xls` worksheet.

Subsystem Name

The subsystem name of the VLE, which is set by the VLE installation scripts, is specified in the following:

- Either the VTCS `CONFIG TAPEPLEX STORMNGR` parameter or the `CONFIG STORMNGR NAME` parameter.

- The VTCS `CONFIG RTD STORMNGR` parameter.

- The SMC `STORMNGR NAME` parameter.

- The SMC `SERVER STORMNGR` parameter.

- The HSC `STORCLAS STORMNGR` parameter.

VTSS Ethernet Port Addresses

The VTSS Ethernet port addresses are required to configure the VTSS to VLE IP connection via the `CONFIG RTD IPIF` parameter. 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 but does **not** correspond to an actual value on the VSM 6 TCP/IP ports.

IP Addresses of VLE Ports for Host (UUI) Communication

These addresses are required for the SMC `SERVER IP` parameter.

VMVC Volsers

Required to define VMVCs to SMC/VTCS, method of definition depends on the software version, see "Defining the VLE VMVCs to the MVS Host Software and Including VMVCs in an MVC Pool" on page 25.

VMVC Reclamation Threshold

For more information, see"Specifying the Reclamation Policy for VMVCS" on page 24.

VTV Deduplication

The `STORCLAS DEDUP` parameter specifies whether VTV data migrated to VMVCs in the specified `STORMNGR` is deduplicated. For example:

```
STORCLAS NAME(VLEDEDUP)STORMNGR(VLE1) DEDUP(YES)
```

This `STORCLAS` statement specifies to deduplicate data in Storage Class `VLEDEDUP` that is migrated to `VLE1`. For more information, see *ELS 7.x Command, Control Statement, and Utility Reference,*

Deduplication increases effective VMVC capacity and is performed by the VLE before the VTV is written to a VMVC. Oracle recommends, therefore, that you initially enable deduplication, then monitor the results with the `SCRPT` report and fine tune deduplication as necessary.

Early Time to First Byte (ETTFB)

ETTFB (also known as the concurrent tape recall/mount feature) allows host applications to read data while VTVs are being recalled from **either** VMVCs or RTDs. ETTFB is done by overlapping the VTV recall and mount phases, which allows the application to read the VTV data sooner. If the application attempts to read part of the VTV that has not been recalled, the application's I/O request is blocked until the required VTV data is been recalled. With ETTFB for VLE, application access to the first byte occurs in under a second, making the VLE a true extension of the VTSS. Therefore, VLE ETTFB is a good choice for applications that serially access the VTV data. VLE ETTFB is generally **not** a benefit for those applications that stack multiple files on a single VTV, including HSM and image management applications. In these types of applications, the desired data is usually not at the beginning of the VTV, but rather at some random location in the VTV.

ETTFB is disabled by default. You can globally enable ETTFB via the `CONFIG GLOBAL FASTRECL` parameter. If you globally enable ETTFB, you can disable it for individual VTSSs via the `CONFIG VTSS NOERLYMNT` parameter.

VTVs that have incurred a ETTFB recall error have an error flag set in their VTV record in the CDS. These VTVS are subsequently not selected for ETTFB. If you want to reset the error flag, do any of the following:

- Enter a `VTVMAINT SCRATCH(ON)` command for the VTV.

- Migrate the VTV to a new MVC copy.

- Import the VTV.

- Create a new version of the VTV.

- Scratch the VTV.

MVS Host Software Configuration Tasks

Adding VLE to a VSM system requires the tasks described in the following sections:

For more information on the commands and control statements referenced in this chapter, see *ELS 7.x Command, Control Statement, and Utility Reference.*

Acquire the ELS supporting PTFs for VLE 1.3

For ELS 7.2, support is included in the base level. For ELS 7.0 and 7.1, get the latest `SMP/E receive HOLDDATA` and PTFs described in TABLE 2-1 and `SMP/E APPLY` with `GROUPEXTEND`.

TABLE 2-1 ELS Supporting PTFs for VLE 1.3

ELS 7.0	ELS 7.1
L1H16C1	L1H16J6
L1H1672	L1H1674

Updating the SMC OMVS RACF Security Entry

The VLE requires SMC to have an OMVS RACF security entry in order to have a TCP/IP connection to the host.

OMVS is a segment associated with the RACF userid. The SMC started task must have a userid associated with OMVS, either in the `RACF STARTED` class definition or the `ICHRIN03 LNKLST` module. The userid associated with the SMC task needs to have an OMVS segment defined to it within RACF as follows:

```
ADDUSER userid
DFLTGRP(groupname)OWNER(owner)OMVS(UID(uidnumber))
```

Or, if the userid already exists but does not have an OMVS segment:

```
ALTUSER userid OMVS(UID(uidnumber))
```

Modifying the SMC SCMDS file

SMC manages all communication between VTCS and VLE, so SMC must know how to connect to the VLE server. You do so by adding an SMC STORMNGR statement for each VLE system plus one or more SMC SERVER statements that define the TCP/IP control paths for the VLE. For 7.0 and above, you may want to do this in your SMC CMDS file as shown in CODE EXAMPLE 2-1.

```
TAPEPLEX NAME(TMVSA)LOCSUB(SLS0)
SERVER NAME(ALTSERV) TAPEPLEX(TMVSA) +
HOSTNAME(MVSX) PORT(8888)
STORMNGR NAME(VLE1)
SERVER NAME(VLESERV1)+
STORMNGR(VLE1)IP(192.168.1.10)PORT(60000)
```

CODE EXAMPLE 2-1 SMC Commands for VLE

CODE EXAMPLE 2-1 contains:

- A TAPEPLEX statement, which defines a single TapePlex, TMVSA, with an HSC/VTCS running on the same MVS host (SLS0).

- A SERVER statement, which defines a backup HSC/VTCS subsystem (ALTSERV) running on another host.

- A STORMNGR command that defines a VLE (VLE1) .

- A second SERVER command that defines a UUI communication path to the VLE, where:

  - The server name is VLESERV1.

  - The STORMNGR parameter value is VLE1.

  - The IP parameter value is the VLE 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 VLE.

Updating the VTCS CONFIG Deck to Define VLE

You must update VTCS `CONFIG` deck to define the VLE and the connectivity from the VTSS systems to the VLE. VTCS can drive VLE 1.3 as follows:

- **For VTCS 7.0** and above, the `CONFIG TAPEPLEX` statement defines the TapePlex that VTCS is running under and provides the list of defined VLEs on the `CONFIG TAPEPLEX STORMNGR` parameter as shown in CODE EXAMPLE 2-2.

```
TAPEPLEX THISPLEX=TMVSA STORMNGR=VLE1
VTSS NAME=VTSS1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTDPATH  NAME=VL1RTD1 STORMNGR=VLE1 IPIF=0A:0
RTDPATH  NAME=VL1RTD2 STORMNGR=VLE1 IPIF=0A:1
RTDPATH  NAME=VL1RTD3 STORMNGR=VLE1 IPIF=0I:0
RTDPATH  NAME=VL1RTD4 STORMNGR=VLE1 IPIF=0I:1
RTDPATH  NAME=VL1RTD5 STORMNGR=VLE1 IPIF=1A:0
RTDPATH  NAME=VL1RTD6 STORMNGR=VLE1 IPIF=1A:1
RTDPATH  NAME=VL1RTD7 STORMNGR=VLE1 IPIF=1I:0
RTDPATH  NAME=VL1RTD8 STORMNGR=VLE1 IPIF=1I:1
VTD LOW=6900 HIGH=69FF
```

CODE EXAMPLE 2-2 VTCS 7.0 CONFIG VLE Example

In CODE EXAMPLE 2-2, note:

- The `CONFIG TAPEPLEX` statement, which defines `TMVSA` as the TapePlex that VTCS is running under and the names of all connected VLEs (which in this example is a single VLE called `VLE1`).

- The `CONFIG RTDPATH` statements, which define a single VLE RTD for each path from the VTSS to the VLE. In this example, the `CONFIG RTDPATH` statements for `VTSS1` specify:

  - The name of the `RTDPATH`.

  - The connections to the defined VLEs (`STORMNGR=VLE1`).

  - The `IPIF` value for each VTSS to VLE port connection in *ci:p* format where:

    - *c* is 0 or 1.

    - *i* is A or I.

    - *p* is 0 through 3.

    Note – 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 but does **not** correspond to an actual value on the VSM 6 TCP/IP ports.

- VTCS 7.1 and above systems can, of course, drive VLE 1.3 as VTCS 7.0 does as described . In this mode, however, the number of VLE RTD targets is limited by the number of paths from a VTSS. Additionally, the VLE RTDs are assigned to fixed VTSS paths. The path from a VTSS to the VLE is always reserved by VTCS regardless of whether any VTSS to VLE data transfer is occurring.

  However, with VTCS 7.1 (with the PTFs described in) and above, you can define a VLE with more VLE RTD targets then there are paths from the VTSS to the VLE, which means:

  - The path from the VTSS to the VLE is **not** reserved unless a VTSS to VLE data transfer is required.

  - More VLE RTD operations can occur simultaneously. For example, an audit of a VMVC does not require data transfer between the VTSS and the VLE.

  As shown in , the VLEs are defined via a CONFIG STORMNGR statement, **not** the CONFIG TAPEPLEX STORMNGR parameter. The CONFIG STORMNGR statement specifies the VLEs that VTCS connects to. Additionally, for each VLE, the CONFIG STORMNGR VLEDEV parameter defines the number and the names of the RTD devices that the VLE emulates. The more devices defined (up to the maximum of 96 devices per VLE), the greater the level of concurrent activities VTCS can schedule on the VLEs.

```
TAPEPLEX THISPLEX=TMVSC
STORMNGR NAME=VLE1 VLEDEV(S000-S05F)
STORMNGR NAME=VLE2 VLEDEV(S000-S05F)
VTSS NAME=VTSS1 LOW=70 HIGH=80 MAXMIG=8 MINMIG=4 RETAIN=5
RTDPATH  NAME=VL1RTD1 STORMNGR=VLE1 IPIF=0A:0
RTDPATH  NAME=VL1RTD2 STORMNGR=VLE1 IPIF=0A:1
RTDPATH  NAME=VL1RTD3 STORMNGR=VLE1 IPIF=0I:0
RTDPATH  NAME=VL1RTD4 STORMNGR=VLE1 IPIF=0I:1
RTDPATH  NAME=VL1RTD5 STORMNGR=VLE2 IPIF=1A:0
RTDPATH  NAME=VL1RTD6 STORMNGR=VLE2 IPIF=1A:1
RTDPATH  NAME=VL1RTD7 STORMNGR=VLE2 IPIF=1I:0
RTDPATH  NAME=VL1RTD8 STORMNGR=VLE2 IPIF=1I:1
VTD LOW=6900 HIGH=69FF
```

CODE EXAMPLE 2-3 VTCS 7.1 CONFIG VLE Example

In , note:

- The CONFIG TAPEPLEX statement now simply defines TMVSC as the TapePlex that VTCS is running under. It does **not** define the connected VLEs.

- The CONFIG STORMNGR statements, which define the VLEs configured in this system - VLE1 and VLE2 in this example. These statements also specify the number of VLE devices via the VLEDEV parameter. In this example, each VLE has the maximum of 96 emulated devices, which allows VTCS to schedule up to 96 processes on each VLE. The VLE device addresses are in the form of S*xxx* (where *xxx* is a hexadecimal value. For example, S000-S05F represents 96 emulated devices.

- The `CONFIG RTDPATH` statements for VTSS1, which specify:

    - The name of the `RTDPATH`.

    - The connections to the defined VLEs (`STORMNGR=VLE1,` `STORMNGR=VLE2`).

    - The `IPIF` value for each VTSS to VLE port connection in *ci:p* format where:

        - *c* is 0 or 1.

        - *i* is A or I.

        - *p* is 0 through 3.

        Note – 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 but does **not** correspond to an actual value on the VSM 6 TCP/IP ports.

Specifying the Reclamation Policy for VMVCS

VLE MVC media (VMVCs) is subject to fragmentation and must be reclaimed just like real MVCs. The VMVC reclaim process, however, uses far fewer resources than a standard reclaim. The reclaim threshold for a VMVC is specified via the `CONFIG RECLAIM VLTHRES` parameter. The lower that you set `VLTHRES`, the more frequent VTCS will run reclaim on the VMVCs and the greater the effective capacity of the VMVS (less fragmentation).

Defining the VLE VMVCs to the MVS Host Software and Including VMVCs in an MVC Pool

VMVC volsers must be defined both to the MVS host software and to the VLE. The VMVCs are defined to the VLE as part of the VLE configuration. The following sections tell how to define the VMVCs to the MVS host software.

▼ Creating VMVC Volume Pools (7.0 and Above)

1. **Code HSC POOLPARM/VOLPARM statements to define the VMVC pools.**

   For example, to define two separate pools for VLE1 and VLE2:

   ```
   POOLPARM NAME(LEPOOL1)TYPE(MVC)
   VOLPARM VOLSER(VL0000-VL880)

   POOLPARM NAME(LEPOOL2)TYPE(MVC)
   VOLPARM VOLSER(VL2000-VL2880)
   ```

2. **Run SET VOLPARM to validate the POOLPARM/VOLPARM statements.**

   ```
   SET VOLPARM APPLY(NO)
   ```

   `APPLY(NO)` validates the statements without loading them. If you like the results, go to Step 3. Otherwise, rework your volume definitions, rerun this step, and if the definitions are valid, then go to Step 3.

3. **Run SET VOLPARM to load the POOLPARM/VOLPARM statements.**

   ```
   SET VOLPARM APPLY(YES)
   ```

Updating the MVS Host Software Policies

The following sections tell how to update the MVS host software policies to direct data to the VLE system.

Creating Storage and Management Classes for VLE

Management Classes specify *how* VTCS manages VTVs. The HSC `MGMTclas` control statement defines a Management Class and its attributes. For example, the `DELSCR` parameter of the `MGMTclas` statement specifies whether VTCS deletes scratched VTVs from the VTSS. Management Classes can also point to *Storage Classes*, which specify *where* migrated VTVs reside. The HSC `STORclas` control statement defines a Storage Class and its attributes. You specify the VLE system as the destination for migrated VTVs via the `STORCLAS STORMNGR` keyword. For example:

```
STOR NAME(VLOCAL) STORMNGR(VLE1) DEDUP(YES)
STOR NAME(VREMOTE) STORMNGR(VL2)DEDUP(YES)
```

The preceding statements define a "local" Storage Class (`VLOCAL`) on the `VLESERV1` and a "remote" Storage Class (`VREMOTE`) on the on the `VLESERV2`. As these `STORCLAS` statements specify, all migrations to storage class `VLOCLAL` or `VREMOTE` **must** go to the specified VLEs. **Also note** that deduplication is specified for both Storage Classes.

You can be less restrictive than this if desired. For example, if you define an MVCPOOL that contains both VMVCs and MVCs you can set up the migration policies to migrate to a VLE but if the VLE becomes full or not available, to continue to migrate to real tape media (MVCs). For example the MVC pool `DR` is defined as follows:

```
POOLPARM NAME(DR)TYPE(MVC)
VOLPARM VOLSER(VL0000-VL0100)
VOLPARM VOLSER(ACS000-ACS099)
```

Pool `DR`, therefore, contains both MVCs and VMVCs. A Storage Class that specifies pool DR will migrate first to VMVCs and only use MVCs if VMVCs are not available. For example:

```
STOR NAME(DRCLASS) MVCPOOL(DR)DEDUP(YES)
```

This method is valuable if you have a configuration where both an ACS and a VLE are connected to the VTSS systems.

Next, to specify migration to VLE, you specify the VLE Storage Classes you defined via the `MGMTCLAS MIGPOL` parameter. For example:

```
MGMT NAME(M1) MIGPOL(VLOCAL,VREMOTE)
MGMT NAME(M2) MIGPOL(DRCLASS)
```

Management Class `M1` migrates one VTV copy to the "remote" VLE, one copy to the "local" VLE. Management Class `M2` migrates a single VTV copy to the Storage Class that points to the "mixed" MVC pool that contains both MVCs and VMVCs.

Note – In addition to directing migration to a VLE also consider:

1. You can use the `ARCHAge` and `ARCHPol` parameters of the MGMTclas statement to set an Archive Policy for VTVs in a Management Class. When the VTV's age exceeds the `ARCHAge` value, the VTV is eligible for archive per the Storage Class(es) specified on the `ARCHPol` parameter. You can use Archive Policies to archive (move) VTVs from VMVCs to MVCs as the VTVs age. For more information, see *Managing HSC and VTCS*.

2. You can use `STORSEL` statements to cause VTCS to preference recalls from VLE media. For more information, see *Managing HSC and VTCS*.

3. If you are running at ELS 7.0 or above, you can use HSC MIGRSEL and MIGRVTV to fine tune migration to VLE. Using these statements, you can cause migration of data in a Management Class to start to one Storage Class before another. This method is typically used to ensure that a critical DR copy is made as soon as possible. For more information, see *Configuring HSC and VTCS.*

4. On a VLE 1.1 and above system, if multiple VLEs are connected to each other and to the VTSS, *by default*, VTCS preferences VLE to VLE connections to make multiple VTV copies. You can control this behavior as described in "FROMLST Parameter Use".

FROMLST Parameter Use

If you do **not** specify FROMLST, the **default** behavior is as follows:

• For Clustered VTSSs, if a copy resides on multiple VTSSs in the cluster, the VTV can be sourced from any available VTSS, which may not be optimal if the VTSS and the connected ACS are geographically distant from each other.

• For VLE to VLE connections, if a VTV copy resides on both a VTSS and one VLE and you want to migrate it to a connected VLE, the default is to use the VLE to VLE connection. Similarly, this may not be optimal if the connected VLEs are geographically distant from each other.

shows a DR scenario with a local VLE (LOCVLE) and remote VLE (REMVLE) connected to VTSSA. You wish to migrate two VTV copies:

- First, a local copy from VTSSA to LOCVLE
- Second, a copy via VLE-to VLE copy from LOCVLE to REMVLE.
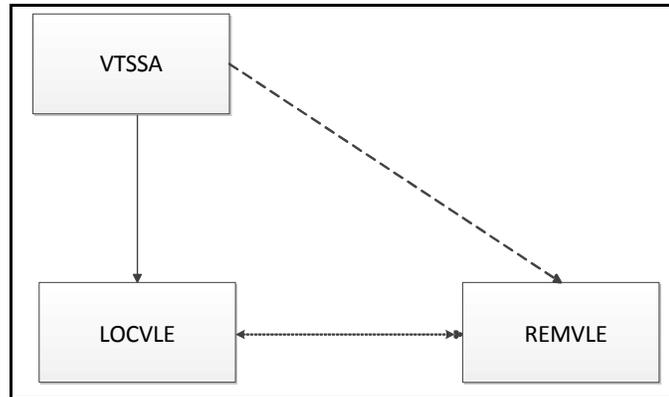
FIGURE 2-1 VLE to VLE Replication (Versus VTSS to VLE)

To make the VTV copies as desired, do the following:

1. **Create a** VTSSLST **statement to create a VTSS list that contains only** VTSSA**.**

   `VTSSLST NAME(VSM2VLE) VTSS(VTSSA)`

2. **Create a** STORCLAS **statement that sends a VTV copy to** REMVLE**.**

   `STORCLAS NAME(FORREMOT) STORMNGR(REMVLE)`

3. **Create a** MIGRVTV **statement that delays the migration copy to** REMVLE**.**

   `MIGRVTV STOR(FORREMOT) IMMDELAY(360)`

   The migration delay is 360 minutes to allow the migration to the local site occur first; then the migration to the remote site is by VLE-to-VLE copy. **Note that** 360 minutes is just an example value, you can specify values of up to 9998 (**do not** specify 9999, because then the VTV is only migrated via automigration).

4. **Create a** STORCLAS **statement that sends a VTV copy to** LOCVLE**.**

   `STORCLAS NAME(FORLOCAL) STORMNGR(LOCVLE) FROMLST(VSM2VLE)`

   Note that the FROMLST parameter specifies that the local VTV copy is sourced from VTSSA.

5. **Finally, create a** MGMTCLAS **statement that specifies two VTV copies, one to the local site and one to the remote site:**

   `MGMTCLAS NAME(DRVLE) MIGPOL(FORLOCAL,FORREMOT)`

Routing Data to VLE

To route data to VLE, first create an SMC POLICY command that specifies a VLE Management Class. Next, create SMC TAPEREQ statements that route the desired workload to the SMC VLE policy. For example

```
POLICY NAME(VLEBACK) MEDIA(VIRTUAL) MGMT(DRVLE)
TAPEREQ DSN(CRITICAL.**) POLICY(VLEBACK)
```

The preceding example assigns the VLEBACK policy to all tape datasets with an HLQ of CRITICAL.