Oracle Video Server™

Administrator's Guide and Command Reference

Release 3.0 for UNIX

February 2, 1998

Part No. A53961-02

ORACLE®

Enabling the Information Age™

Oracle Video Server Administrator's Guide and Command Reference

Part No. A53961-02

Release 3.0

Printed in the U.S.A

Primary Author: Nancy Baltz, Christian Bedford

Contributors: Kim Bartlett, Mark Brewster, Mike Christian, Susie Ehrsam, Ming Lee, Alex Lind, Parker Lord, Dave Pawson, Prakash Ravulaparti, Pat Ritto, David Robinson, Andrew Samuels, Tom Sepez, Shawn Shyh, Manish Upendram

Contents

1 Tasks of the Oracle Video Server Administrator

2 System Planning for the Oracle Video Server

3 Basic Oracle Video Server Configuration Tasks

4 Starting and Stopping Oracle Video Server

5 Monitoring and Maintaining Log Files

6 Monitoring Clients

7 Oracle Media Data Store Tasks and Procedures

10 Configuring the Session-and-Circuit Service

11 Oracle Video Server Components

12 Oracle Media Data Store (MDS) Utilities

13 Stream Service Utilities

14 Session-and-Circuit Service Utilities

Figures

Tables

Send Us Your Comments

Oracle Video Server Administrator's Guide and Command Reference: UNIX, Release 3.0 Production

Part No. A53961-02

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- electronic mail - omsdoc@us.oracle.com
- FAX - 650.506.7615 Attn: Oracle Media Server Documentation Manager
- postal service:
  Oracle Corporation
  Oracle Media Server Documentation
  500 Oracle Parkway, Mailstop 6OP5
  Redwood Shores, CA 94065
  U.S.A.

If you would like a reply, please give your name, address, and telephone number below.

_____

_____

_____

Preface

This *Oracle Video Server Administrator's Guide and Command Reference* documents operational procedures and provides a description of all server processes and command-line utilities for the Oracle Video Server (OVS). Before you begin, read *Introducing Oracle Video Server* for a general overview of Oracle Video Server and how it works.

This preface discusses:

About This Manual

This guide describes system administration tasks required to operate and maintain Oracle Video Server operating on a UNIX server platform, including:

- system planning
- starting and stopping Oracle Video Server
- monitoring system processes
- configuring services

This guide is divided into the following parts and chapters:

Part I: Oracle Video Server Administrator's Guide describes how to operate and configure your Oracle Video Server system. Chapters in Part I are:

- Chapter 1, "Tasks of the Oracle Video Server Administrator"outlines the tasks required to operate and configure Oracle Video Server.
- Chapter 2, "System Planning for the Oracle Video Server" covers the decisions you must make before you begin an installation of Oracle Video Server.
- Chapter 3, "Basic Oracle Video Server Configuration Tasks" describes basic server configuration tasks.
- Chapter 4, "Starting and Stopping Oracle Video Server" describes how to start and stop Oracle Video Server.
- Chapter 5, "Monitoring and Maintaining Log Files" describes how to configure the message logging system and view log files.
- Chapter 6, "Monitoring Clients" describes how to run utilities and examine log files to monitor the Oracle Video Server software.
- Chapter 7, "Oracle Media Data Store Tasks and Procedures" explains how to maintain and expand the Oracle Media Data Store (MDS) file system.
- Chapter 8, "Using Real-time Video Feeds" describes how to use and configure the Real-time Feed Service.
- Chapter 9, "Configuring the Logical Content and Scheduling Services" describes how to create a database account for Oracle Video Server, and how configure the Logical Content and Scheduling Services.
- Chapter 10, "Configuring the Session-and-Circuit Service" describes how to configure Oracle Video Server for different network environments.

Part II: Oracle Video Server Command Reference provides details on all of a given command's options and capabilities. Chapters in Part II are:

- Chapter 11, "Oracle Video Server Components" describes Oracle Video Server's services and configuration files.
- Chapter 12, "Oracle Media Data Store (MDS) Utilities" describes the utilities used to perform file operations for the MDS file system.
- Chapter 13, "Stream Service Utilities" describes utilities used to register content files, convert certain file types into formats recognized by Oracle Video Server, and perform database operations on files.
- Chapter 14, "Session-and-Circuit Service Utilities" describes utilities for monitoring and disconnecting active clients.

System-Specific Information

Information in this guide generally applies to all UNIX systems unless a specific attribution is given. Details particular to one system are noted in the text of this guide. For information specific to your server operating system, please refer to the *Oracle Video Server Installation Guide* for your server platform.

Conventions Used in This Manual

Table 1 defines the typographical conventions used in this guide.

Convention	Explanation
bold	Identifies filenames, command names, and system variables when discussed in the normal text.
italics	Identifies new terms and system variables when discussed in the normal text.
`monospace width`	Identifies command syntax and system output.
`monospace bold`	Highlights parts of command syntax or output specific to the discussion about that command or output.

Table 1: Conventions

Convention	Explanation
monospace italic	Identifies variables in examples for which a context-specific substitution should be made. For example, the variable *filename* would be replaced by an actual filename.
[ ]	Identifies optional items.
[a|b]	Identifies a choice of optional items, each separated by a vertical bar (|); any one option can be specified. In this example you may select either **a** or **b**, but not both.
{a|b}	Identifies a choice of mandatory items, each separated by a vertical bar (|). In this example you must select either **a** or **b**.
%	Identifies the C-shell prompt and is used in examples.
#	Identifies the prompt for the user **root** (also referred to as the **superuser**).
$	Identifies environment variables.
...	Indicates that the preceding item can be repeated any number of times.
(online only)	Denotes a document available only in HTML or Adobe Acrobat format.

Table 1: Conventions

Example Conventions

This guide shows example commands in this font:

```
% mdscopy oracle1.mpg /mds/video/oracle1.mpg
```

For an example command that is longer than a single line, a backslash (\) appears at the end of a line to indicate the command continues on the next:

```
% mdstar -c -b 128 -f /gfs/dev/scsi/rst053 \
/mds/video/oracle2.mpg /mds/video_archive/oracle2.mpg
```

Note Do not enter the backslash on your command line. It is used here as a typographic convention.

Operating System Conventions

This guide uses commands from UNIX C-shell. In this text of this guide, UNIX keywords appear in **boldface** and UNIX parameters appear in *italics*.

Related Documents

For a list of related documents, see the *Oracle Video Server Documentation Roadmap*.

Audience

This document is for database administrators and others responsible for installing Oracle products on UNIX operating systems. While command examples are provided, this document does not attempt to teach Oracle or UNIX system administration.

Your Comments Are Welcome

We value and appreciate your comments as an Oracle user and reader of the manuals. As we write, revise, and evaluate our documentation, your opinions are the most important input we receive. At the back of our printed manuals is a Reader's Comment Form, which we encourage you to use to tell us what you like and dislike about this manual or other Oracle manuals. If the form is not available, please use the following address or FAX number.

Oracle Video Server Documentation Manager
Oracle Corporation
500 Oracle Parkway
Box MS 6op5
Redwood City, CA 94065
U.S.A.
FAX: 650-633-3045
Email: omsdoc@us.oracle.com

Part I

Oracle Video Server Administrator's Guide

1

Tasks of the Oracle Video Server Administrator

This chapter introduces tasks specific to administering Oracle Video Server, including.

- System Planning
- Basic Configuration Tasks
- Starting and Stopping Oracle Video Server
- Using the Message Logging Service
- Monitoring Clients
- Maintaining the Oracle Media Data Store
- Using the Real-time Feed Service
- Configuring the Logical Content and Scheduling Services
- Configuring the Session-and-Circuit Service

System Planning

Oracle Video Server administrators are responsible for creating a real-time video system that meets their organization's needs. This includes creating a system that can deliver the required number of concurrent video streams, creating additional Oracle Media Data Store (MDS) volumes as the organization's multimedia library grows, and scaling the system appropriately as multimedia content and demand increases.

For detailed information on system planning, refer to Chapter 2, "System Planning for the Oracle Video Server."

Basic Configuration Tasks

To provide optimal performance, you may need to adjust some server configuration parameters upon completing the installation of the Oracle Video Server software. Basic configuration parameters include:

- adjusting the maximum number of streams. The number of streams Oracle Video Server provides sets the number of concurrent users who can access video and audio programming.
- assigning video pumps to processors to improve system performance
- starting multiple video pump processes
- configuring Oracle Video Server to use low bit rate video. By providing video at a lower bit rate, you can reduce the amount of network bandwidth required to deliver video.
- specifying a configuration file for use by the session and circuit service. The circuit and session configuration file allows Oracle Video Server to support more complex network configurations, and improves network load balancing

Chapter 3, "Basic Oracle Video Server Configuration Tasks" describes the basic configuration tasks.

Starting and Stopping Oracle Video Server

Oracle Video Server provides two user interfaces:

- Oracle Video Server Manager (VSM), a JAVA-based, management console which provides an easy to use graphical display from which to operate Oracle Video Server
- Command line utilities and scripts from which to configure and operate Oracle Video Server using the UNIX command shell.

The VSM console is an optional tool, providing an easy to use graphical interface. This is the preferred interface from which to operate Oracle Video Server, however, you may operate Oracle Video Server from the UNIX command shell with command line utilities and configuration scripts. The interface from which you choose to operate Oracle Video Server determines the startup procedure you use.

Chapter 4, “Starting and Stopping Oracle Video Server” describes how to start and stop Oracle Video Server using either the VSM console or the UNIX command shell.

> **Note:** The VSM console is intended as a tool for regular operational tasks, content management, and system monitoring. Some Oracle Video Server features must be configured and operated from the command line to provide optimal performance.

Using the Message Logging Service

Oracle Video Server’s message logging service allows you to track informational and error messages pertaining to the system’s performance and operation. Careful monitoring of message logs can help you predict and identify the sources of system problems. For example, if log messages show that the video pump is failing to deliver video on time, you may need to reduce the number of concurrent users your system supports, or increase hardware resources for the system to function properly.

The message logging service has several configuration options allowing you to:

- log specific events and event types
- assign a logging service to a specific process. This allows you to more easily monitor a specific aspect of Oracle Video Server’s operation
- specify a message log file type. You can configure the logging service to write messages to one of several log file types, including ASCII text, binary, and the system console.

- archive log files.

Chapter 5, “Monitoring and Maintaining Log Files” describes how to view log files and configure the message logging service.

Monitoring Clients

You can monitor client activity and server resources in use by clients using the VSM console or command line utilities. By monitoring user demand on your system you can better plan system upgrades and isolate potential problems. Oracle Video Server also provides a command line utility to disconnect clients. Before shutting down the server for maintenance or other administrative tasks you can remove clients.

Chapter 6, “Monitoring Clients” describes how to monitor client activity and disconnect active clients from the server.

Maintaining the Oracle Media Data Store

The Oracle Media Data Store (MDS) file system provides real-time access to video and audio data. As your video library grows, you will need to add additional volumes to the MDS, rename volumes and, on occasion, replace failed disk drives.

Chapter 7, “Oracle Media Data Store Tasks and Procedures” provides detailed information on performing these and other file system tasks.

Using the Real-time Feed Service

Oracle Video Server’s Real-time Feed Service allows you to:

- encode a video feed from a live source and prepare it for almost immediate playback
- encode and prepare video programming for playback in one-step

This feature allows broadcasters to provide live programs in near real-time with the added advantage that viewers can tune in late and view a program from the beginning, or pause an event and resume viewing from where they left off. You can also use the Real-time Feed Service to encode your video and prepare it for playback in one step.

Chapter 8, “Using Real-time Video Feeds” describes how to configure and use the Real-time Feed Service.

Configuring the Logical Content and Scheduling Services

Oracle Video Server provides two features which require the use of an external database. The features are:

- Logical Content Service
- Scheduling Service

The Logical Content Service

The Logical Content Service provides an abstract view of content files stored in the MDS file system. The Logical Content Service does not work with the files stored in the MDS directly, but with metadata describing the MDS files stored in a database. Using the Logical Content Service provides greater control and flexibility both in managing video and audio data and in building multimedia applications.

The Scheduling Service

The Scheduling Service provides a mechanism broadcasters can use to schedule logical content titles for playback on specific channels. This service can be used for television broadcasting, near video-on-demand, pay-per-view television, and LAN multicasts.

Chapter 9, “Configuring the Logical Content and Scheduling Services” describes how to create a database account for Oracle Video Server and configure the Logical Content and Scheduling Services.

Configuring the Session-and-Circuit Service

The Session-and-Circuit Service is responsible for managing server resources for clients. A client session maintains the mapping between the different components of Oracle Video Server on behalf of the client. You can create a configuration file mapping instances of the video pump to client groups which you define. Such a configuration file improves network load balancing.

Chapter 10, “Creating a Session-and-Circuit Configuration File” describes how to create a configuration file for your network environment.

2

System Planning for the Oracle Video Server

Successfully providing digital video and audio programs to users across a network requires careful planning. This chapter is designed to inform you about the performance and scalability issues involved in networked multimedia, and the decisions you must make to successfully deploy Oracle Video Server within your organization.

The purpose of this chapter is to help you plan an Oracle Video Server installation which meets the needs of your organization's users. Topics in this chapter are:

Oracle Video Server Planning Issues

Oracle Video Server is an end–to–end software solution for networked client / server systems which store, manage, deliver, and play digitally encoded video and audio. To successfully deploy an Oracle Video Server system, first consider several interrelated issues which are highlighted in this chapter:

- **How many users must concurrently access video programming?**

  The user base you plan to concurrently serve effects several factors, including: the number of processors the server must contain, the sustained bandwidth of the network used to distribute video from Oracle Video Server to users, and the amount of sustained throughput the MDS file system must be able to provide.

- **What quality of video and audio programming do you intend to provide**?

  In general, the higher quality video required, the greater the encoding rate which must be used to digitize the video files.

- **How many video files will you need to store? How much disk space will be required to store the video files?**

  The number of files you need to store and the disk capacity required to store the files effects how you plan the MDS file system.

- **Must the system be available 24 hours a day, seven days a week?**

  If so, you must consider a system with parity protected data. By using parity protection, Oracle Video Server can continue providing video to user's even if a disk fails.

As you read this chapter, think about these issues and how they relate to your organization. By carefully considering your organization's needs, you can implement an Oracle Video Server system which meets the needs of your users and allows for future growth.

Encoding Rates and Concurrent Users

This section discusses trade-offs among the number of video streams to be provided, the quality of that video, and the system resources available.

Choosing an Encoding Rate

The number of video streams that Oracle Video Server can concurrently provide, and the quality of the video, are interrelated. To provide high quality video, the video source must be encoded at a relatively high bit rate (2.0 Mbps for example) which requires more server throughput and network bandwidth than video

encoded at a lower rate (1.2 Mbps). If a greater number of concurrent video streams are required, using the same computing resources, you will need to lower the encoding rate to a value which places less bandwidth demand on both the Oracle Media Data Store (MDS) disk system and the network interface being used.

While video encoding is beyond the scope of this document, you should consider the quality of video you require and the encoding rate needed to provide that level of service. To learn more about video encoding, refer to the *Oracle Video Server Administrator's Content Guide*.

Server Hardware Considerations

Oracle Video Server's performance and scalability directly relates to the server hardware in use and the configuration of the server software. To provide continuous, streaming media to users across a network requires a server computer with adequate processing power, system I/O, and memory. This section discusses server hardware and its effect on Oracle Video Server's performance.

Using Uni- or Multi-processor Servers

The more processors your server hardware contains, the greater the scalability of your Oracle Video Server system. The video pump, the server process responsible for sending video data from the server to users across a network, can provide programs to a greater number of concurrent users when operating on servers with more than one processor. **Note that at least one processor must always be left available to run the server's operating system, the remaining Oracle Video Server processes, and any other applications running on the server.**

Table 2–1 Ratio of processors to pumps

Number of processors	Number of video pumps you can start
1	1 video pump
2	1 video pump assigned to a processor
3	2 video pumps, each assigned to a processor
4	3 video pumps, each assigned to a processor

When operating in a production environment, use a server with two or more processors. This allows you to assign a single video pump process to a processor,

thereby improving scheduling and performance. For development systems providing programs to only a few concurrent users, a single processor server may be acceptable.

Memory Requirements

The amount of available memory your server has affects two interrelated issues:

- the number of concurrent video streams the system can provide
- the bit rate of the video streams

If you intend to provide programs to a large user base, or provide programs encoded at a relatively high bit rate, consider installing more memory. The *Oracle Video Server Installation Guide* cites the minimum amount of memory the server platform must have to run Oracle Video Server.

Disk Controllers

A factor which greatly affects server performance and scalability is disk I/O. To provide media programs to several users at once, you must create a disk array capable of sustained throughput adequate for your user's needs. Ensure that the server hardware supports an adequate number of disk controllers for your installation.

You will find more information on planning and creating the Oracle Media Data Store (Oracle Video Server's file system) in *Planning the Oracle Media Data Store* on page 2-7.

Network Interface Cards

To provide adequate network load balancing and bandwidth, consider using a server which supports multiple network interface cards. The greater the number of network cards, the more concurrent users your installation can service.

You will find more information on computer networks appropriate for video and audio delivery in *Understanding Network Bandwidth Requirements* on page 2-5.

Understanding Network Bandwidth Requirements

The network bandwidth available to the server determines the number of concurrent video streams Oracle Video Server can provide. For example, while a Fiber or Copper Distributed Data Interface (FDDI /CDDI) provides a maximum throughput rate of 100 Mbps, it is the sustained throughput rate which is important for video delivery. Sustained throughput is generally 60–80% of the maximum rate, or 60–80 Mbps in the case of FDDI or CDDI. The throughput of the server's network interface can severely limit the number of concurrent video streams and/or the encoding rate used.

To determine the number of concurrent video streams your server's network interface can support at a given bit rate, use the following formula:

$$\frac{\textit{throughput of network interface}}{\textit{bitrate of video stream}} = \textit{concurrent video streams}$$

where:

throughput of network interface	the sustained transfer rate of the server's network interface in Mbps
bit rate of video streams	the encoding rate of the video in Mbps
concurrent video streams	the number of concurrent video streams that can be provided

Example 2–1 Calculating network bandwidth

If the network interface provides sustained throughput of 60 Mbps, and the video is encoded at 1.50 Mbps, 40 concurrent video streams can be provided.

$$\frac{60\ Mbps}{1.50\ Mbps} = 40\ concurrent\ streams$$

To ensure that the server can meet the bandwidth demands required by the number of concurrent video streams to be provided, consider a server with support for multiple network interfaces, and confirm that the server is able to support the aggregate transfer rates for all attached devices.

Network Packet Size

> **Note:** This information applies to local-area networks using UDP or TCP/IP. ATM or broadband networks may use another packet size depending on user requirements and network configuration.

When operating in local-area networks using UDP or TCP/IP, Oracle Video Server uses a network packet size of 8K to transmit video data. Depending on the network protocol and components in use, these packets may be fragmented further as they are routed from source to destination.

To ensure proper delivery of video data from Oracle Video Server to client devices, test your network with the **ping** utility. Use **ping** to send packets of 8 K between your client device and the server running Oracle Video Server.

Example 2–2 Checking your network with ping

```
C:\WINDOWS>ping oracle-video -l 8192
Pinging [144.25.56.75] with 8192 bytes of data:
Reply from 144.25.56.75: bytes=32 time<10ms TTL=253
Reply from 144.25.56.75: bytes=32 time<10ms TTL=253
Reply from 144.25.56.75: bytes=32 time<10ms TTL=253
Reply from 144.25.56.75: bytes=32 time<10ms TTL=253
```

This example uses **ping** from an MS-DOS prompt to check the network connection to the server **oracle-video**. The packet size to use is specified with the **-l** option. Note that the packet size is specified in bytes. To specify a packet size of 8K, use 8192 bytes.

If you are unable to transmit 8K packets between your system's client devices and the server running Oracle Video Server, verify that all of the networking software and equipment (TCP/IP stacks, routers, switches, and network cards) are capable of handling 8K packets.

For more information on creating a multimedia capable network, refer to your network vendor's documentation.

Planning the Oracle Media Data Store

The Oracle Media Data Store (MDS) is a file system designed to store multimedia data and deliver it in real-time to multiple clients. In order to successfully implement Oracle Video Server in your organization, give thought to the following key factors in determining the configuration of the MDS:

- bit rate of the video content
- RAID size
- amount of content to be stored
- number of concurrent video streams to be provided

Bit Rate of Encoded Video and Storage Capacity

Video content files must be encoded before they can be loaded into the MDS and made available for playback. Appropriate MPEG-1 encoding rates for video delivery when using OVS on a corporate Local Area Network (LAN) range from 1.366 to 2.048 Mbps. The higher the bit rate of the encoded content, the more MDS storage that is required. The following formula calculates storage requirements for different encoding rates:

$$\textit{storage requirements (MB)} = \frac{\textit{total play length (seconds)} \times \textit{bit rate (Mbps)}}{\textit{8 bit/byte}}$$

where:

storage requirements	the amount of storage space required to hold the digitized audio and video content files in megabytes
total play length	the total length of time in seconds of all video and audio files combined
bit rate	the bit rate at which the content is encoded in megabits per second, for example 1.36 Mbps
8 bit/byte	converts Mbps (megabits per second) to MB (megabytes)

Every content file that you load into the MDS must be registered. This process creates a tag file for each content file. When calculating how much storage space you need, add approximately 10% to the content storage requirements to account for tag files.

Examples

The following examples store 10,000 video clips, each with an average length of 30 seconds, encoded at 1.366 and 2.048 Mbps respectively:

Storage for video encoded at 1.366 Mbps:

First, calculate the storage space used by each video clip:

$$\textit{storage requirements (MB)} = \frac{\textit{30 seconds} \times \textit{1.366 Mbps}}{\textit{8 bit/byte}} = \textit{5.12 MB}$$

Finally, multiply the storage requirements of each video clip by the number of clips to be stored. This example also adds 10% to the storage requirements to account for tag files:

$$\textit{10, 000} \times \textit{5.12 MB} \times \textit{1.10} = \textit{56.3 GB of storage}$$

To store 10,000 video clips with an average length of 30 seconds, encoded at 1.366 Mbps, and the associated tag files, will require 56.3 GB of disk space.

Storage for video encoded at 2.048

Calculate the storage space used by each video clip:

$$\textit{storage requirements (MB)} = \frac{\textit{30 seconds} \times \textit{2.048 Mbps}}{\textit{8 bit/byte}} = \textit{7.68 MB}$$

Multiply the storage requirements of each video clip by the number of clips to be stored. This example also adds 10% to the storage requirements to account for tag files:

$$\textit{10, 000} \times \textit{7.68 MB} \times \textit{1.10} = \textit{82.5 MB of storage}$$

To store 10,000 video clips with an average length of 30 seconds, encoded at 2,048 Mbps, and the associated tag files, requires 82.5 GB of disk space.

Protecting the Data

The MDS provides two mechanisms which allow data to remain available in the event of disk failures:

- software RAID support
- spare disks

Software RAID Support

The MDS file system uses RAID (Redundant Array of Inexpensive Disks) technology to improve both performance and fault-tolerance. The implementation of RAID used by Oracle Video Server is volume striping with parity (RAID 5).

All files stored in the MDS file system are striped across the disks in the volume. When files are written to stripe sets, the data is broken into stripes, which are sequentially written to all drives in the volume. As a result, a file is spread across all drives in the volume. Striping data across multiple disks and disk controllers greatly improves read / write performance. For example, when you write a file to the MDS, the first block of the file is written to the first available block on the first disk of the volume; the next block of the file is written to the next available block on the second disk of the volume, and so on until the file has been completely written across all available disks. This is much faster than writing sequentially to the same disk. The same is true when reading data from the volume. Thus, the effective access time for a striped volume is substantially less than the time that would be required to access individual drives.

To protect your data, the MDS file system uses *parity*. Stripe sets with parity write data sequentially across disks, but also create a parity block for each stripe on a given RAID set. The parity information is used to reconstruct data in the event of a single disk failure within a given RAID set.

By striping files across disk drives and including parity information, a fault-tolerant disk array with minimal drive overhead is created. For example, a common method of providing fault-tolerant arrays is *disk mirroring*. Disk mirroring uses two drives that are maintained as mirror images; that is each drive contains the same data as the other. A pair of mirrored drives loses 50 percent of total drive capacity to achieve fault tolerance. A stripe set with parity consisting of four drives loses just 25 percent of total capacity to create parity records. A stripe set with five drives loses only 20 percent of total capacity to create a fault-tolerant configuration.

Note: The Oracle Video Server installation gives you the option of installing Oracle Video Server without parity protection. Oracle recommends the use of volume striping with parity protection for all production systems. Volume striping without parity protection is suitable only for development systems, where the availability of data does not affect users viewing programs.

Using a Spare Disk

Oracle Video Server supports the use of a spare disk, which can be used if a single disk drive fails within an MDS volume. If the MDS volume uses parity protection, the data from a failed disk drive can be reconstructed to the spare. This ensures that your MDS volume can remain operational should another hard disk fail before you can replace the failed disk. One spare disk can be used per MDS volume.

Spare disks are an optional feature of Oracle Video Server, but are recommended if you wish to maintain high availability of data.

Note: If your disk array supports disk hot-swapping, you do not need not use a spare disk.

Choosing a RAID size

The first step in creating an MDS volume is choosing a *raidsize*. The raidsize is the number of disks in each RAID set (See Figure 2–1). Consider these issues when selecting a raidsize:

- A smaller raidsize increases the number of RAID sets, thereby decreasing the chance of multiple failures in a single RAID set. However, a smaller raidsize also increases the amount of parity information, increasing the portion of the MDS volume storing redundant data and decreasing the available storage space for video data.

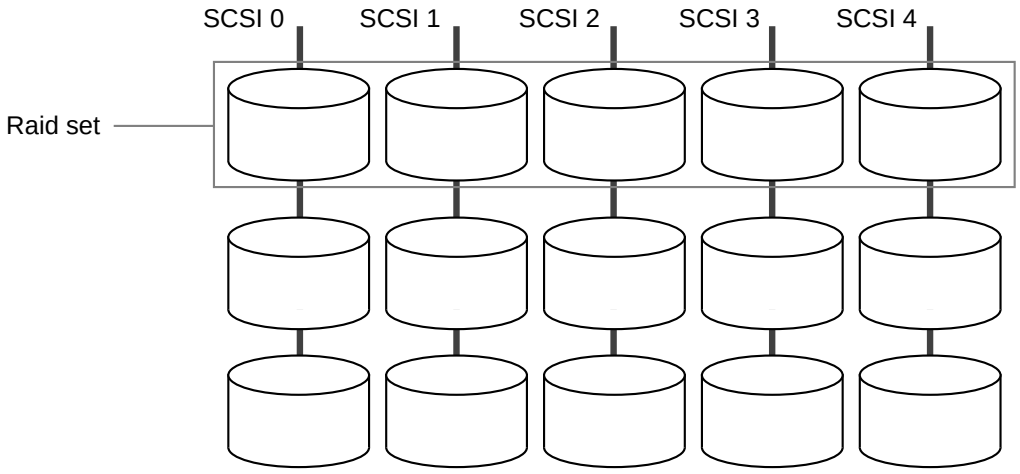
- The raidsize also determines how much memory is consumed by any processes that access the MDS, and should not be set high if memory is limited. The larger the raidsize, the more memory used.
- The raidsize should be equal to or a divisor of the number of SCSI buses used by the MDS so that each SCSI bus supports the same number of disks. This provides fault tolerance in the event of a disk controller failure, and ensures the balanced use of bandwidth across all disk buses.

Note: To ensure data protection, you must always stripe data **across** disk chains. Never place more than one disk on any chain in the same RAID set. Failing to do this will result in data loss should a disk drive or disk controller fail.

Do not stripe data down disk chains! Doing so will result in data loss should a disk drive or disk controller fail.

Recommended raidsizes are 3, 4, 5, and 6. In most cases a raidsize of 5 is recommended. A raidsize of 5 provides an acceptable balance between the chance of multiple failures in one RAID set and the portion of parity data stored (20 percent). Figure 2–1 shows an MDS volume with three RAID sets of five disks each, making the raidsize five. Each SCSI bus supports three disks.

Figure 2–1 A typical MDS volume configuration with 3 RAID sets.



The required disk space can be calculated with the following formula:

$$disk\ space = \frac{content \times raidsize}{raidsize - 1}$$

where:

disk space	The amount of disk space required to accommodate both your content and RAID protection.
content	The amount of content to be stored in GB.
raidsize	The number of disks in each RAID set.
raidsize-1	Subtracting one from the raidsize accounts for parity data.

The following examples calculate storing 82.5 Gbs of content using a raidsize of 4, 5, and 6. The total storage requirements for the video clips, including the tag files, is calculated as in the previous formula:

Using a raidsize of 4

$$\frac{82.5\ GB \times 4}{4 - 1} = 110\ GB$$

Using a raidsize of 5

$$\frac{82.5\ GB \times 5}{5 - 1} = 103\ GB$$

Using a raidsize of 6

$$\frac{82.5\ GB \times 6}{6 - 1} = 99 GB$$

Choosing the Number of Disk Drives

The number of disk drives required for an MDS volume depends on:

- the amount of data you need to store
- the number of disks in each RAID set
- the capacity of the disks
- the disk bandwidth (I/O) required to support the number of concurrent users

> **Note:** For information on supported disk configurations and capacities, refer to the *Oracle Video Server Installation Guide* for your server platform.

Based on the required storage capacity, use this formula to calculate how many disks your MDS volume needs:

$$disks = \frac{disk\ space}{disk\ size} + spares$$

where:

disk space is the disk space required by your MDS volume as calculated in the previous formula.

disk size is the storage capacity of a single disk. Several smaller disks (2 to 4 GB) provide better performance than a few larger disks.

spares is the number of spare disks you are designating for your MDS volume. Spare disks are used to replace a failed disk while the MDS is running if your hardware does not support hot swapping.

If you are using RAID, round up the calculated number of disks to the nearest multiple of the RAID size so that all RAID sets in the MDS volume are complete. Since spare disks are not part of a RAID set, be sure to round up the number of disks before adding the number of spares.

Storage Examples

The following example calculates the number of 4 GB disks required to store 82.5 GB of content using a RAID size of 5. The storage capacity for the given RAID size is from a previous example:

$$\frac{82.5\ GB \times 5}{5 - 1} = 103\ GB$$

To store 103 GB of data using 4 GB disks, you must use 30 disks (25.75 rounded to the nearest multiple of the RAID size is 30). The resulting volume configuration would be 6 RAID sets each with a RAID size of 5 for a total storage capacity of 120 GB.

$$\frac{103\ GB}{4\ GB} = 25.75\ disks$$

Choosing a Stripe Width

The MDS supports 32K or 64K stripe widths. The stripe width is the amount of data (in kilobytes) written to a disk stripe. The stripe width you choose effects the amount of disk throughput and the amount of memory used to cache data. A smaller stripe width results in less disk throughput, but requires less memory in which to cache data. A larger stripe width results in greater disk throughput, but requires more memory in which to cache data.

The Oracle Video Server installation allows you to choose either a 32 KB or 64 KB stripe width. Oracle recommends a stripe width of 32 KB for most systems.

Number of Files to Store

The number of files the MDS can store is limited by the number of RAID stripes reserved for the MDS volume's Table of Contents (TOC). The default MDS configuration reserves only one RAID stripe for the TOC and uses 64 bytes for each file stored. The following formula calculates the maximum number of files that an MDS volume can store:

$$number\ of\ files\ =\ \frac{stripe\ width \times 1024\ bytes \times raidsize}{64\ bytes\ per\ file} - 1$$

where:

stripe width	Size in KBs of the RAID stripe. In general this value should be 32 K.
raidsize	The number of disks in each RAID set. If you are using parity protection, subtract 1 to account for the duplicated data.
64 bytes per file	size in bytes of the TOC entry used to store information for a file.

The table below lists the number of entries each TOC stripe will store using a stripe width of 32K and raidsizes of 3 to 6 with parity protection enabled.

Table 2-1 Number of TOC entries

Stripe width	raidsize-1	Number of files each TOC stripe can store
32	3	1,023 files
32	4	1,535 files
32	5	2,047 files
32	6	2,559 files

When calculating the number of stripes to reserve for the MDS volume's TOC, you must also account for the tag files created for each video or audio file. For example, if you have 2,000 individual video files, you will have 2,000 tag files. Thus the total number of files the MDS must store is 4,000.

Specifying the TOC Size

The number of stripes to reserve for TOC entries is set using the ***tocsz*** parameter of the **voltab** file. Before initializing an MDS volume, ensure that the ***tocsz*** parameter is set to an appropriate value. For example, is you are using an MDS volume with a raidsize of six, the default ***tocsz*** will allow you to store 2,559 files (see Table 2-1). If you need to store 5,000 files (2,500 content files and 2,500 tag files), you must specify that the MDS volume reserve two stripes for the TOC.

For information on the ***tocsz*** parameter and the **voltab** file, refer to the **voltab** section of Chapter 11, "Oracle Video Server Components."

Planning Bandwidth Capacity for User Demand

The MDS file system plays an important role in determining the number of concurrent streams that can be provided by Oracle Video Server. The MDS uses a value for the maximum bandwidth (also called the bandwidth capacity) available for all Oracle Video Server components to both read from and write to the MDS volume. If this parameter, called **maxbw**, is not sufficient, it can restrict the number of concurrent media streams available.

This section tells you whether the *bandwidth capacity* of your MDS volume can meet the *bandwidth demand* on your MDS volume.

- The *bandwidth demand* on your MDS volume is based on the number of concurrent video streams you plan to serve from the MDS volume, and the bit rates of these streams.
- The *bandwidth capacity* is the smaller of these values:
  - the total disk bandwidth, based on the number of disks in the MDS volume and the bandwidth of a single disk
  - the total SCSI bus bandwidth, based on the number of buses in the MDS volume and the sustained bandwidth of each bus

This section also describes:

- how to calculate bandwidth demand and bandwidth capacity
- how to increase bandwidth capacity if it does not meet demand

Calculating Bandwidth Demand

Bandwidth demand is equal to the number of concurrent video streams the MDS volume must serve, multiplied by the bit rate of these streams, plus approximately 20% to allow for concurrent MDS volume loading and other administrative access. The formula to calculate bandwidth demand is:

bandwidth demand = *video streams* x *bitrate* x ((*1 + 20%*))

where:

video streams	is the number of concurrent video streams you plan to serve from the MDS volume.
bit rate	is the bit rate of these streams (for example, 1.3 Mbps).

Examples Relating Bandwidth Demand to Concurrent Streams

The following examples highlight how the bandwidth demand relates to both the number of video streams provided by Oracle Video Server, and the rate at which the video content is encoded.

Example 2–1 40 concurrent video streams encoded at 1.3 Mbps

If you require 40 concurrent video streams encoded at 1.3 Mbps, the **maxbw** of the MDS volume must be at least 62.4 Mbps:

bandwidth demand = 40 concurrent streams $\times$ *1.3 Mbps* (*1 + 20%*) = *62.4 Mbps*

Example 2–2 40 concurrent video streams encoded at 2.0 Mbps

If you require 40 concurrent video streams encoded at 2.0 Mbps, the **maxbw** of the MDS volume must be at least 96 Mbps.

bandwidth demand = 40 concurrent streams $\times$ *2.0 Mbps* (*1 + 20%*) = *96 Mbps*

Example 2–3 80 concurrent video streams encoded at 2.0 Mbps

If you require 80 concurrent video streams encoded at 2.0 Mbps, the **maxbw** of the MDS volume must be at least 192 Mbps.

bandwidth demand = 80 concurrent streams $\times$ *2.0 Mbps* (*1 + 20%*) = *192 Mbps*

Bandwidth Capacity: Total Disk Bandwidth

The total disk bandwidth for your MDS volume is equal to the number of disks in the MDS volume (not including spares) multiplied by the bandwidth of an individual disk adjusted by a factor to account for RAID protection:

$$\textit{total disk bandwidth} = \frac{\textit{disks} \times \textit{disk bandwidth} \times (\textit{raidsize} - 1)}{\textit{raidsize}}$$

disks is the number of disks in your MDS volume, not including spares.

disk bandwidth is the sustained bandwidth of the disks in your MDS volume. The bandwidth is a function of the stripe width you specify when you define the MDS volume and the speed of the disk hardware. Conservative estimates for bandwidth are:

- 1.0 Mbps for a stripe width of 32K
- 1.5 Mbps for a stripe width of 64K

raidsize is the number of disks in each RAID set. Subtract 1 to account for the parity data distributed among the RAID set's disks.

Example 2–4

If you have eight disks, a stripe width of 32K, and a RAID size of four, your total disk bandwidth is approximately 48 Mbps. Note the conversion factor between megabytes (MBps) and megabits (Mbps):

$$\textit{total disk bandwidth} = \frac{8 \times 1.0\ \textit{MBps} \times (8\textit{Mbps}/\textit{MBps}) \times (4 - 1)}{4} = 48\ \textit{Mbps}$$

Bandwidth Capacity: Total SCSI Bus Bandwidth

The total SCSI bus bandwidth for your MDS volume is equal to the number of SCSI buses in the MDS volume, multiplied by the sustained bandwidth of an individual bus, adjusted by a factor to account for RAID:

$$\textit{total bus bandwidth} = \frac{\textit{buses} \times \textit{sustained bandwidth} \times (\textit{raidsize} - 1)}{\textit{raidsize}}$$

where:

buses	is the number of buses in your MDS volume. Each bus is associated with a chain of SCSI disks.
sustained bandwidth	is the estimated sustained bandwidth for your SCSI buses, which is about 60% of the peak bandwidth. Table 2–1 shows both the peak and average bandwidths of different SCSI bus types

Table 2–1 SCSI Bus Peak Bandwidths

SCSI Type	Maximum Bus Bandwidth	Sustained Bus Bandwidth (Approx.)	Maximum Disks per Bus
Ultra-Wide	40 MB / sec	20 MB / sec	15
Fast-Wide	20 MB / sec	12 MB / sec	15
Fast-Narrow	10 MB / sec	6 MB / sec	7
Single-Ended	5 MB / sec	3 MB / sec	7
The bandwidth estimates in this table are conservative. The actual bandwidth of your disks may vary depending upon their SCSI type and size.			

Table 2–1 also shows the maximum number of disks on the chain associated with each type of bus. Ensure that the SCSI card used by your server is capable of supporting the maximum number of drives you require (for example, although the Fast-Wide SCSI standard allows 15 disks per bus, not all vendors' buses support that many disks).

raidsize is the number of disks in each RAID set.

Follow these guidelines when choosing a SCSI type:

- Because of the difference in bandwidths, use Fast-Wide or Ultra-Wide SCSI and avoid Single-Ended SCSI if possible.
- Choose the same type of bus throughout the MDS volume. If you use different types of buses, the total bus bandwidth is limited by the slowest bus.

Calculating SCSI Bus Bandwidth

The following examples calculate bandwidth for different SCSI bus types and raidsizes.

Example 2–5

If you have four Fast-Wide SCSI buses and a raidsize of four, your total SCSI bandwidth is 288 Mbps. Note the conversion factor between MBps and Mbps:

$$\textit{total bus bandwidth} = \frac{4(2)(60\% \times 20MBps)(8\ Mbps/MBps)(4-1)}{4} = 288\ Mbps$$

Example 2–6

This example uses six Fast-Wide SCSI buses and a raidsize of 6 to produce a SCSI bandwidth of 480 MBps.

$$\textit{total bus bandwidth} = \frac{6(60\% \times 20MBps)(8\ Mbps/MBps)(6-1)}{6} = 480\ Mbps$$

As you can see, increasing the raidsize greatly increases the amount of available bandwidth. Table 2-3 lists bandwidths for different raidsizes using Fast-Wide and Ultra-Wide SCSI buses.

Table 2–2 SCSI bus bandwidths

raidsize	Fast-Wide	Ultra-Wide
3	192 Mbps	384 Mbps
4	288 Mbps	576 Mbps
5	384 Mbps	768 Mbps
6	480 Mbps	960 Mbps

Note: Your system's SCSI bandwidth may vary from the values provided here. The bandwidth values above do not take backplane throughput or operating system limitations into consideration. For information on obtaining accurate bandwidth values for your system, refer to Obtaining Accurate Bandwidth Values later in this chapter.

Adjusting the MDS Volume Bandwidth

Compare your bandwidth demand with your total disk bandwidth and your total bus bandwidth. If both the total disk bandwidth and the total bus bandwidth are greater than or equal to your bandwidth demand, your MDS volume can reliably serve the planned number of streams. When you define the MDS volume in the **voltab** file, use **mdsvolstat** to confirm the bandwidth empirically, and set the ***maxbw*** parameter to that value.

If your bandwidth demand exceeds either the total disk bandwidth or the total bus bandwidth, your MDS volume cannot reliably serve the planned number of streams.

To increase the total bandwidth capacity, follow these measures:

- To increase the total disk bandwidth:
  - Use more disks.
  - Increase the RAID size. Note the trade-offs discussed in Choosing a RAID size earlier in this chapter.
- To increase the total bus bandwidth:
  - Use more SCSI buses and redistribute your available disks among them.
  - Use a SCSI type with a greater sustained bandwidth.
  - Increase the RAID size. Note the trade-offs discussed in Choosing a RAID size earlier in this chapter.

Obtaining Accurate Bandwidth Values

To obtain the ***maxbw*** value of your MDS volume, use the **mdsvolstat** utility. The **mdsvolstat** utility determines the ***maxbw*** value by doing multiple data transfers to and from the MDS disks. The final number it returns is based on the physical I/O performance of the volume. For more information on **mdsvolstat**, refer to Chapter 12, "Oracle Media Data Store Utilities."

3

Basic Oracle Video Server Configuration Tasks

Once you have installed Oracle Video Server, you may wish to undertake some basic configuration tasks to enhance performance and scalability for your user environment. This chapter discusses basic configuration tasks to help you maximize the performance of your Oracle Video Server system. Topics in this chapter are:

Using the Administrative Interfaces

Oracle Video Server provides two administrative interfaces from which you can configure the system. The interface you choose depends upon the level of configuration you plan to implement on your system. The administrative interfaces are:

- Oracle Video Server Manager
- UNIX command line

Oracle Video Server Manager

Oracle Video Server Manager (VSM) provides a graphical user interface through which you can manage and monitor Oracle Video Server. In addition to operational tasks, the VSM console provides an interface from which you can perform basic configuration tasks. Note that the VSM console allows you to configure systems using only one video pump process. If your system adequately supports your organization's user base using a single video pump, you may use VSM as your primary configuration tool.

Note: You must create a UNIX group under which to install and operate Oracle Video Server. Only primary members of the group account you create can log into VSM. Ensure that all users performing administrative tasks via VSM are primary members of the UNIX group which owns Oracle Video Server.

UNIX Command Line

If you require a more complex configuration, using multiple video pump processes to support your user base, you must configure Oracle Video Server by editing the **ovsstart** file. The **ovsstart** file serves as both a startup script and configuration file for Oracle Video Server services and processes.

Note: If you use **ovsstart** to configure Oracle Video Server, you can still perform administrative tasks using the VSM console. The only functionality disabled in VSM when using the **ovsstart** file is the ability to modify start up preferences using the VSM console.

Table 3-1 delineates the configuration tasks you can perform using both VSM and the UNIX command line, and the configuration tasks that must be performed by modifying the **ovsstart** script.

Table 3–1 System Configuration Tasks Available from VSM and the OVS Command Line

Task	Video Server Manager	OVS Command Line
Changing the maximum number of streams for a single video pump	✓	✓
Changing the video pump packet size	✓	✓
Changing the maximum bit rate per stream	✓	✓
Starting multiple video pump processes		✓
Any video pump configuration task involving multiple video pumps		✓
Specifying a network configuration file	✓	✓

Note: For more detailed information about how to configure Oracle Video Server from VSM, refer to the Oracle Video Server Manager online Help.

This chapter describes how to configure Oracle Video Server by editing the **ovsstart** file. To learn about using VSM to configure Oracle Video Server, refer to:

- *Getting Started with Oracle Video Server Manager*
- Oracle Video Server Manager online Help

About ovsstart

The **ovsstart** configuration file is a plain text file located in the directory path:

```
$ORACLE_HOME/vs30/admin/ovsstart
```

where $ORACLE_HOME is the environment variable that defines the base installation directory for the Oracle Video Server software. For example, if the Oracle Video Server software is installed in the directory /usr/oracle, the $ORACLE_HOME variable must be set to /usr/oracle.

To find out what value an environment variable is set to, use the **echo** command:

```
% echo $ORACLE_HOME
/usr/oracle
```

Looking at a Sample Configuration File

This section shows an abridged version of the **ovsstart** script. This script shows only the service entries you edit to modify Oracle Video Server's configuration settings. The omitted sections of the script check to see if any Oracle Video Server services are already running, and that Oracle Video Server's environment variables are correctly set. **Do not modify these sections of the script**.

You may need to change the values or options used by a particular service (such as the number of video streams provided by the video pump), modify the start-up script to start the service with the required values. For more information on the services, and the command line options accepted by each service, refer to the *Oracle Video Server Command Reference* in Part II of this guide.

Note: Before modifying the **ovsstart** file, create and archive a backup copy of the file.

Here is an annotated version of the **ovsstart** configuration file. Startup scripts vary from system to system, so do not be concerned if the script for your system does not exactly match the example provided here:

```
echo -n "starting media data store ....... "
$ORACLE_HOME/bin/mdsdirsrv -W -f $ORACLE_HOME/vs30/admin/voltab &
sleep 5
```

The **mdsdirsrv** (MDS directory server) process manages files in the MDS file system and controls access to those files. This example uses the **-W** option to mount all MDS volumes specified in the **voltab** file in read-write mode, allowing new files to be written to the MDS while users access files for playback.

```
echo -n "starting remote server....... "
$ORACLE_HOME/bin/mdsrmtsrv &
sleep 5
```

The **mdsrmtsrv** (MDS remote file server) process enables remote access to the MDS.

```
echo -n "starting session manager ....... "
$ORACLE_HOME/bin/vscsmsrv &
sleep 1
```

The Session and Circuit Service (**vscsmsrv**) maps physical addresses of client devices to physical downstream addresses on the server computer. You can enhance network bandwidth and load balancing by creating a configuration file mapping client addresses to network interfaces and video pumps. You will find information on creating a configuration file in Chapter 10, "Configuring the Session-and-Circuit Service."

```
echo -n "starting video pump ....... "
$ORACLE_HOME/bin/vspump -a -o pumpA -m 10 -n 8192 -b 3100000 -t &
sleep 3
```

The video pump (**vspump**) streams content in real time from the MDS to one or more clients over a network. This example configures the video pump to provide 10 concurrent media streams (specified with the **-m** option) at a maximum bit rate of 310000 bps (specified with the **-b** option). The default video pump installation uses a packet size of 8192 bytes (specified with the **-n** option), which is used in LAN-based video deployments.

Note: This example also uses the **-t** option. The **-t** option improves the video pump's real-time scheduling and performance when running on Sun Solaris operating systems. This option should only be specified when running Oracle Video Server on the Sun Solaris operating system.

The **ovsstart** file includes two additional video pump configurations that you can implement by removing the comment characters and editing the configurations to suit your Oracle Video Server deployment.

```
#   Option:
#   To assign a dedicated cpu to vspump, use the -y switch with cpu number.
#   For example:
#
# echo -n "starting video pump, lock to cpu 1 ....... "
# $ORACLE_HOME/bin/vspump -a -o pumpA -m 5 -n 8192 -b 3100000 -t -y 1 &
# sleep 3
#
```

The first option uses the **-y** option to lock a video pump to a processor. This provides increased performance and allows better video pump scheduling. When running Oracle Video Server on multiprocessor servers, you can start multiple video pumps and lock each to a processor (minus one processor to run the server operating system and other Oracle Video Server processes). To learn more about the video pump and its configuration options, see Chapter 11, "Oracle Video Server Components."

```
#   For modem connections, recommended low bitrate output buffer size is 1k.
#   For example:
#
# echo -n "starting video pump, with low bitrate configuration ....... "
# $ORACLE_HOME/bin/vspump -a -o pumpA -m 5 -n 1024 -b 3100000 -t &
# sleep 3
```

The second option uses a packet size of 1024 bytes (specified with **-n**) to deliver video streams in low bit rate environments.

```
echo -n "starting stream service ....... "
$ORACLE_HOME/bin/vsstrmsrv -n 30 &
sleep 1
```

The stream service (**vsstrmsrv**) handles requests for content files from clients accessing the server. When the stream service receives a request for a specific content file, it reads a tag file associated with the requested file in the MDS. The stream service then coordinates delivery of the content with a video pump. This example specifies (with the **-n** option) that the stream service accept up to 30 concurrent clients. The default when **-n** is not specified is 20 concurrent connections.

When adjusting the number of streams provided by the video pump, you must adjust the number of concurrent connections the stream service will accept to the same value. A single **vsstrmsrv** process can service multiple video pumps, however, you must adjust the value of the **-n** option to be equal to or greater than the combined number of streams specified with the video pump's **-m** option.

Note: On systems delivering hundreds of video streams, it may be necessary to start multiple instances of **vsstrmsrv**. This improves load balancing and scalability.

To learn more about setting the number of streams and user connections, see Setting the Number of Concurrent Users on page 3-10.

```
echo -n "starting content service ....... "
$ORACLE_HOME/bin/vscontsrv &
sleep 1
#   Option:
#   To use without logical content, implement both of the following:
#   1) Omit -c command line option
#   2) Omit vscontsrv.connect resource from the
#      $ORACLE_HOME/vs30/admin/mnrc file
#
#   To use with logical content, implement one of the following:
#   1) Startup vscontsrv without -c command line option
#      or
#   2) Include vscontsrv.connect=username/password@connect_string resource
#      from the $ORACLE_HOME/vs30/admin/mnrc file
#
echo -n "starting content service ....... "
$ORACLE_HOME/bin/vscontsrv &
sleep 1
```

The content service (**vscontsrv**) enables a client device to query the MDS for a listing of available content. The **vscontsrv** process can query either the MDS file system for a list of available files to play or it can use logical content titles stored in a database. For information on the Logical Content Service, refer to Chapter 9, "Configuring the Logical Content and Scheduling Services."

```
#   Option:
#   To startup scheduling services,
#   use the following.  Note:
#   1) Logical Content Service must be enabled, and
#   2) Include the -c command line switch or add the
#      vsbcastsrv.connect resource specification to
#      the $ORACLE_HOME/vs30/admin/mnrc file #
# echo -n "starting broadcast initiator .. "
# $ORACLE_HOME/bin/vsbcastsrv &
# sleep 5
# echo -n "starting broadcast data service........ "
# $ORACLE_HOME/bin/vsnvodsrv &
# sleep 2
# echo -n "starting scheduler ............ "
# $ORACLE_HOME/bin/vsschdsrv &
# sleep 2
```

The Scheduling Service allows you to play programs according to a schedule on a network channel you specify. This service is intended for broadcasters, and consists of the following processes:

- **vsbcastsrv**
- **vsnvodsrv**
- **vsschdsrv**

You will find more information on the Scheduling Service in Chapter 9, "Configuring the Logical Content and Scheduling Services."

```
echo -n "starting MDS FTP Server ....... "
$ORACLE_HOME/bin/mdsftpsrv -w -p 1621 &
sleep 1
```

The MDS FTP service (**mdsftpsrv**) enables an FTP client to transfer binary files to or from the MDS file system. This example starts **mdsftpsrv** with **-w**, giving FTP access to any authorized user on the system. The port specified with **-p** is 1621, the port recommended by Oracle. To coexist with other FTP daemons, **mdsftpsrv** must use a specific port on which to "listen" for FTP requests.

```
#   Option:
#   To use the Real-Time Feed option
#
# echo -n "Starting Real-Time Feed service ....."
# $ORACLE_HOME/bin/vsfeedsrv &
# sleep 5
```

The Real-time Feed Service (**vsfeedsrv**) enables:

- a live broadcast to be digitally encoded, loaded into the MDS file system, and prepared for immediate playback
- encoding and loading of video files in one step, eliminating the process of individually copying and registering files with the MDS file system.

You will learn more about the Real-time Feed Service in Chapter 8, "Using Real-time Video Feeds."

Basic Configuration Tasks

The basic configuration tasks are:

- **Setting the number of concurrent users.**

  You can determine the maximum number of users that can connect to Oracle Video Server. This allows you to configure the server appropriately for the amount of disk and network bandwidth available.

- **Adjusting the Maximum Bit Rate**

  Specifying a bit rate slightly higher than that of the video programs you intend to play allows the video pump to better allocate system resources. For this reason you should adjust the maximum bit rate the video pump will provide based on the encoding rate of your content.

- **Configuring the video pump for optimum performance on multi-processor server hardware.**

  If you operate Oracle Video Server on a server with more than one processor, you must assign (or lock) the video pump process to a processor for its exclusive use. This better guarantees real-time performance.

  If the server hardware has three or more processors, you can start two or more video pump processes and assign each instance to a processor. Starting multiple video pump services improves real-time performance and the ability to provide programs to a greater number of concurrent users.

- **Configuring Low Bit Rate Video Streams**

  You can provide video and audio programs to clients on networks with bandwidth restrictions by configuring Oracle Video Server for low bit rate streams.

- **Creating a configuration file for the Session-and-Circuit Service.**

  The configuration file maps video pumps (and other server resources) to specific network addresses. Creating this mapping file allows you to use multiple network interfaces, increasing network bandwidth and the ability to provide programming to more concurrent users.

Setting the Number of Concurrent Users

You can configure the number of concurrent users that can connect to Oracle Video Server (i.e. the number of users who can connect at the same time). Note that this value is dependent upon your server's hardware configuration. Specifying an

unrealistically high number of user connections may result in video that glitches or users being denied access. For this reason, ensure that the server hardware running Oracle Video Server is adequate for your user needs. Chapter 2, “System Planning for the Oracle Video Server” discusses system planning issues.

You must modify two service entries in the **ovsstart** file to change the number of simultaneous user connections:

- **vsstrmsrv** The stream service controls user requests to Oracle Video Server.
- **vspump** The video pump service streams media files to users. This service determines the maximum number of users that can concurrently receive data from Oracle Video Server.

Configuring the Stream Service

The stream service (**vsstrmsrv**) controls the maximum number of user connections Oracle Video Server allows. The number of user connections is specified with the command option **-n** *integer*. For example, to enable up to 30 concurrent connections, specify 30 as the argument to the **-n** option:

```
$ORACLE_HOME/bin/vsstrmsrv -n 30
```

If **-n** is not specified, **vsstrmsrv** defaults to 20 simultaneous connections.

Configuring the Video Pump

The video pump (**vspump**) controls the maximum number of users who can concurrently receive video programs. The maximum number of users is specified with the option **-m** *integer*. For example, to enable the video pump to deliver data to 30 concurrent users, specify 30 as the argument to the **-m** option:

```
$ORACLE_HOME/bin/vspump -m 30
```

Typical entries for **vspump** and **vsstrmsrv** appear as:

```
echo -n "starting video pump
$ORACLE_HOME/bin/vspump -a -o pumpA -m 10 -n 8192 -b 3100000 &
sleep 3
echo -n "starting content service ....... "
$ORACLE_HOME/bin/vsstrmsrv &
sleep 1
```

This example provides media streams to ten clients simultaneously. The **vsstrmsrv** service allows 20 simultaneous connections by default while the **vspump** service provides media streams to ten clients (**-m 10**).

Adjusting the Maximum Bit Rate

The video pump allows you to adjust the maximum bit rate that it will provide to clients. Since the value you specify can affect server performance, always specify that the video pump provide a maximum bit rate only slightly greater than the video files you intend to play.

Example 3–1 Adjusting the maximum bit rate

```
$ORACLE_HOME/bin/vspump -a -o pumpA -m 10 -n 8192 -b 210000 &
```

To adjust the maximum bit rate, modify the bit rate value specified with the **-b** option. This example adjusts the maximum bit rate to 210000 bps (2.1 Mbps). This would be a suitable bit rate with which to deliver video files encoded at up to 2.0 Mbps.

Configuring Video Pumps for Multi-Processor Servers

A server with two processors allows you to assign (or lock) the video pump process to a processor, better guaranteeing real-time performance. A server with three or more processors allows you to both start multiple video pump processes and assign each to a processor. **Note that at least one processor must always be left available to run the server's operating system, the remaining Oracle Video Server processes, and any other applications running on the server.**

Table 3–2 Ratio of processors to pumps

Number of processors	Number of video pumps you can start
1	1 video pump
2	1 video pump assigned to a processor
3	2 video pumps, each assigned to a processor
4	3 video pumps, each assigned to a processor

Assigning a Video Pump to a Processor

To assign (or lock) a video pump to a processor, use the **-y** option. The **-y** option allows you to assign a processor number, where the first processor contained by the server is labeled zero (0). The syntax of the **-y** option is:

```
vspump -y processor-id
```

Example 3–2 Assigning a video pump to a processor

```
echo -n "starting video pump
$ORACLE_HOME/bin/vspump -a -o pumpA -m 10 -n 8192 -b 3100000 -y 1 &
sleep 3
```

This example assign the video pump process to processor 1.

Example 3–3 Starting multiple video pumps

```
echo -n "starting video pump
$ORACLE_HOME/bin/vspump -a -o pumpA -m 10 -n 8192 -b 3100000 -y 1 &
sleep 3
echo -n "starting video pump
$ORACLE_HOME/bin/vspump -a -o pumpB -m 10 -n 8192 -b 3100000 -y 2 &
sleep 3
```

To start multiple video pump services, create an entry for each video pump service you wish to start. Assign each video pump to a processor with the **-y** option. This example starts two video pumps assigned to processors one and two respectively.

Note the **-o** option used to label each video pump service. The **-o** option allows you to assign a unique label to each video pump instance. When starting multiple video pumps, each video pump must have a unique label.

Configuring Low Bit Rate Video Streams

You can provide video and audio programs to clients on networks with bandwidth restrictions by configuring Oracle Video Server for low bit rate streams.

Note: To provide video and audio programs to users with low bit rate connections, the video and audio files must be encoded at a bit rate appropriate for the connection speed. You will find information on encoding video and audio programs at low bit rates in the *Oracle Video Server Content Administrator's Guide*.

Configuring the Video Pump for Low Bit Rate Streams

To provide video and audio programs at low bit rate, edit the video pump entry to use a packet size (also called a buffer size) of 1024 bytes (1K).

To change the packet size, edit the **-n** option of the video pump process. By default, Oracle Video Server installs the video pump to provide packet sizes for LAN-based clients:

Example 3–4 Configuring the video pump for low bit rate streams

```
$ORACLE_HOME/bin/vspump -a -o pumpA -m 10 -n 1024 -b 128000 &
```

To provide video streams at a low bit rate, change the **-n** option to 1024 bytes. This example also edits the **-b** option, limiting the maximum bit rate the video pump will deliver to 128,000 bps.

Configuring the Session and Circuit Service

The session and circuit service (**vscsmsrv**) is responsible for mapping client network address to downstream addresses maintained by Oracle Video Server. You can create a configuration file mapping server resources to specific network segments, thereby improving network load balancing. By default, Oracle Video Server installs without a configuration file. You can operate Oracle Video Server without a configuration file using the services's built-in defaults, however, you can improve network load balancing and server resource allocation by creating a configuration file for your network environment.

Specifying a Configuration File

To specify that the **vscsmsrv** service use a configuration file, use the option **-f file:** *filename*, where *filename* is the directory path to the configuration file. For example, to specify the file $ORACLE_HOME/vs30/admin/vcm.cfg as the configuration file, edit the **vscsmsrv** entry in the **ovsstart** file shown in Example 3–5.

Example 3–5 Specifying a session and circuit configuration file

```
$ORACLE_HOME/bin/vscsmsrv -f file:$ORACLE_HOME/vs30/admin/vcm.cfg
```

Chapter 10, "Configuring the Session-and-Circuit Service" describes how to create a configuration file for your network environment.

4

Starting and Stopping Oracle Video Server

Oracle Video Server provides two interfaces from which you can perform administrative tasks. The interface from which you choose to operate Oracle Video Server determines how you start the system. This chapter describes start up procedures for the following administrative interfaces:

- Oracle Video Server Manager
- UNIX command line

Using Oracle Video Server Manager to monitor and manage your system does not prohibit you from using the UNIX command line. You can use both interfaces concurrently.

> **Note:** Before operating Oracle Video Server, read the *Oracle Video Server Release Notes* supplied with your release of the software. The *Release Notes* will inform you of any known problems or peculiarities you might encounter when operating Oracle Video Server.

Topics in this chapter are:

Starting Oracle Video Server with the VSM Console

If you wish to use the VSM console to monitor and manage Oracle Video Server, you must first start the Oracle Video Server Manager HTTP Service. To start Oracle Video Server using the Oracle Video Server Manager administrative interface:

1. Log into the server computer with the user account created to administer Oracle Video Server.
2. Establish **$ORACLE_HOME/vs30/admin** as your current working directory:

   ```
   % cd $ORACLE_HOME/vs30/admin
   ```

3. Execute the script **ovsstart** using the option **-w**:

   ```
   % ./ovsstart -w
   ```

   The **ovsstart** script starts Oracle Media Net and Oracle Video Server Manager's HTTP Service. When the HTTP Service is successfully running, the command shell displays output similar to that shown below, and returns the shell to the command prompt:

   ```
   Information: Listening on NORM port 9613 address 0.0.0.0
   Information: The server started successfully
   Server now running as process 29880
   video-srv:
   ```

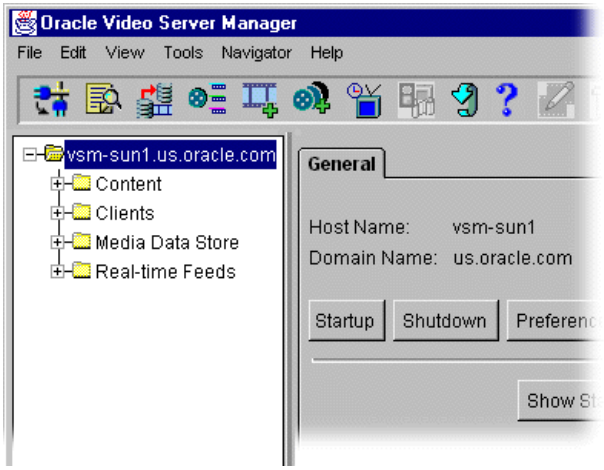
4. Once you have successfully started Oracle Media Net and Oracle Video Server Manager, establish the directory **$ORACLE_HOME/vsmc30/admin** as your current working directory and execute the file **vsmstart**:

   ```
   % cd $ORACLE_HOME/vsmc30/admin
   % ./vsmstart
   ```

5. Oracle Video Server Manager's logon screen appears. Enter your user name and password to log on and click OK. The VSM console starts.

6. From the VSM console, click the **Startup** button. A dialog box will notify you once Oracle Video Server is sucessfully running.

Figure 4–1 VSM console Startup button



To learn more about Oracle Video Server Manager, refer to:

- *Getting Started with Oracle Video Server Manager*
- Oracle Video Server Manager online Help

Starting Oracle Video Server with the Command Line

If you wish to use only the UNIX command line to monitor and manage Oracle Video Server:

> **Note:** If you choose to start Oracle Video Server without Oracle Video Server Manager's HTTP Service, you cannot use the VSM console to perform administrative tasks. If you wish to use both command line and VSM administrative intefaces to operate Oracle Video Server, start the server as described in the section, Starting Oracle Video Server with the VSM Console.

1. Log into the server computer with the user account created to administer Oracle Video Server.
2. Establish **$ORACLE_HOME/vs30/admin** as your current working directory:

   ```
   % cd $ORACLE_HOME/vs30/admin
   ```

3. Execute the shell script **ovsstart**:

   ```
   % ./ovsstart
   ```

What Gets Started?

Oracle Video Server consists of the following server processes:

- OMN address server (**mnaddrsrv**)
- OMN RPC name server (**mnrpcnmsrv**)
- OMN name server (**mnnmsrv**)
- OMN object request broker daemon (**mnorbsrv**)
- OMN logger process **(mnlogsrv)**
- OMN event server (**yeced**)
- MDS directory server (**mdsdirsrv**)
- MDS remote file server (**mdsrmtsrv**)
- session and circuit manager (**vscsmsrv**)
- video pump (**vspump**)

- stream service (**vsstrmsrv**)
- content service (**vscontsrv**)
- MDS FTP server (**mdsftpsrv**)

In addition to these processes, the following optional services may also start depending on the configuration you choose during installation of the Oracle Video Server software:

Scheduling Service

- broadcast data service (**vsbcastsrv**)
- exporter service (**vsnvodsrv**)
- scheduler service (**vsschdsrv**)

Real-time Feed Service

- real-time feed service (**vsfeedsrv**)

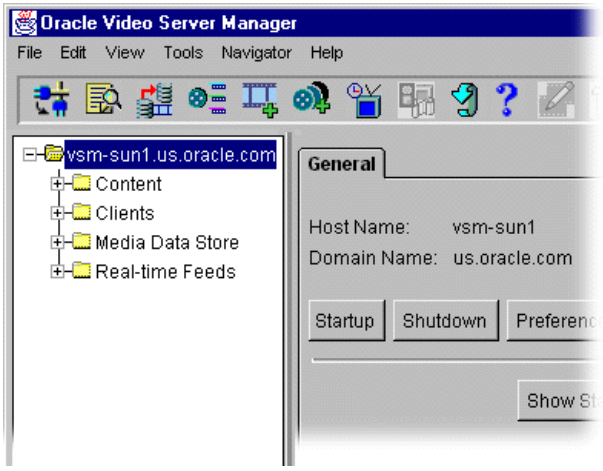
HSM Service

- HSM server (**mdshsmsrv**)
- HSM transfer server (**mdsxfrsrv**)

Stopping Oracle Video Server using the VSM Console

1. Stop Oracle Video Server by clicking the **Shutdown** button on the VSM console. This stops all Oracle Video Server processes except for Oracle Media Net and Oracle Video Server Manager's HTTP Service.

Figure 4–2 VSM console Shutdown button



The shutdown leaves Oracle Media Net and Oracle Video Server Manager's HTTP Service running, allowing you to start Oracle Video Server from the VSM console without having to re-start Oracle Media Net and Oracle Video Server Manager. If you need to shut down the server hardware, or perform other administrative tasks requiring you to stop all applications running on the server hardware, proceed to Step 2.

2. Establish **$ORACLE_HOME/vs30/admin** as your current working directory:

```
% cd $ORACLE_HOME/vs30/admin
```

3. Execute **ovsstop** using the **-a** option to stop all Oracle Video Server processes:

```
% ./ovsstop -a
```

Stopping Oracle Video Server using the UNIX Command Line

1. Establish **$ORACLE_HOME/vs30/admin** as your current working directory:

   ```
   % cd $ORACLE_HOME/vs30/admin
   ```

2. Execute the script **ovsstop**:

   ```
   % ./ovsstop
   ```

Starting Oracle Media Net

Oracle Media Net (OMN) enables communication among Oracle Video Server processes. You will occasionally need to start Oracle Media Net alone to perform various administrative tasks. Typical examples are:

- initializing MDS volumes
- changing an MDS volume's name
- replacing failed disk drives

For information on these and other tasks related to MDS, refer to Chapter 7, "Oracle Media Data Store Tasks and Procedures."

To start Oracle Media Net:

1. Log into the server as the user you established to administer Oracle Video Server.
2. Establish **$ORACLE_HOME/mn33/admin** as your current working directory:

   ```
   % cd $ORACLE_HOME/mn33/admin
   ```

3. Execute the shell script **mnstart**:

   ```
   % ./mnstart
   ```

   This script starts the following Oracle Media Net processes:

   - OMN address server (**mnaddrsrv**)
   - OMN RPC name server (**mnrpcnmsrv**)
   - OMN name server (**mnnmsrv**)
   - OMN object request broker daemon (**mnorbsrv**)

- OMN logger process (**mnlogsrv**)
- OMN event server (**yeced**)

Stopping Oracle Media Net

To stop Oracle Media Net:

1. Log into the server as the user you established to administer Oracle Video Server.
2. Establish **$ORACLE_HOME/mn33/admin** as your current working directory:

   ```
   % cd $ORACLE_HOME/mn33/admin
   ```

3. Execute the shell script **mnstop**:

   ```
   % ./mnstop
   ```

To learn more about Oracle Media Net, see the *Oracle Media Net Administrator's Guide.*

5

Monitoring and Maintaining Log Files

The message logging service records Oracle Video Server *events* in a log file. An event is a significant occurrence in the Oracle Video Server software's operation that requires users to be notified. By monitoring events you can predict and identify potential problems with your Oracle Video Server system. The message logging service consists of the message logging server (**mnlogsrv**) and the log reader utility (**mnlogreader**). The message logging server records event and error messages to a log file. You can read events with either the log reader utility or from the VSM console.

This chapter discusses:

- Configuring the Message Logging Service
- Viewing Log Files

Configuring the Message Logging Service

Oracle Video Server's message logging service (**mnlogsrv**) receives event and error messages from Oracle Video Server services and writes them to a log file which you specify. The **mnlogsrv** service is very flexible in its configuration options, allowing you to:

- define message filters to log specific events and event types
- assign a private logging service to a specific service. Logging services started in this way are not visible to processes other than the specified service.
- specify a log file type such as the system console or the server's logging service (**syslog**)
- archive event logs

To configure **mnlogsrv**, edit the **mnlogsrv** entry of the configuration file $ORACLE_HOME/mn33/admin/mnstart.

Setting the Logging Level

You can define the severity of events recorded by the message logging service by specifying a severity level with the **-f** option of **mnlogsrv**. The severity levels are described in Table 5–1.

Table 5–1 Message logging levels

Severity levels	Description of severity levels
0	Panic condition
1	A condition that should be corrected immediately
2	Critical conditions.
3	Errors
4	Warning messages
5	Non-error conditions that may require special handling
6	Informational messages

Example 5–1 Setting the Severity Level

```
mnlogsrv -f "maxsev 6" -m $ORACLE_HOME/vs30/mesg -m $ORACLE_HOME/mn33/mesg
-p 1024 -o $ORACLE_HOME/mn33/log/mzlog -t textfile
```

This example sets the severity level to "maxsev 6" as indicated by the boldface text. This means that all message levels up to 6 will appear in the message log. To

change the severity level of logged events edit the **-f** option with a level you wish to log.

Note that the higher the logging level you specify, the greater the number of messages written to the log file. This can effect server performance and / or scalability. Choose a logging level which provides the level of information needed to troubleshoot your Oracle Video Server system.

Setting the Message Log File Type

Oracle Video Server's message logging service provides several log file types (or formats). The log file type you choose depends on both your server's operating system and the scalability requirements placed on the server. Table 5-2 lists the log file types and their descriptions.

Table 5–2 Log file types

File Type	Description
tty	Writes error messages to the terminal.
console	Writes error messages to the server's console.
textfile	Writes error messages to an ASCII text file.
binfile	Writes error messages to a binary file.
syslog	Writes error messages to the syslog service in use by the server operating system.

Understanding Log File Types

Most server installations use either an ASCII text or binary log file. Each file type has advantages and disadvantages.

Text file log files are can be opened in a word processor and easily read. Such log files are useful if you wish to share log files with others for their information, or wish to include event logs in documents about Oracle Video Server's operation and behavior in your organization's installation. A disadvantage of text log files is that the message logging service requires more of a server's resources to write the messages to the log. Text log files can also become very large as information is written to them.

Using a binary file (binfile) to record messages is more efficient because messages are recorded using a code instead of being written out ASCII text. Because of this a binary log file uses less system resources, improving server performance and scalability. Binary data also requires less disk space to record the same amount of

information that a text log file requires. A disadvantage of binary log files is that they can only be viewed using the log reader utility (**mnlogreader**).

Specify the message log file type with the **-t** *filetype* option of **mnlogsrv**.

Example 5–2 Specifying a Message Log Type

```
mnlogsrv -f "maxsev 6" -m $ORACLE_HOME/vs30/mesg -m $ORACLE_HOME/mn33/mesg
-p 1024 -o $ORACLE_HOME/mn33/log/mzlog -t textfile
```

This example specifies a **textfile** message log file type.

Archiving Error Logs

Because new records are appended to the log file for each event (error or transaction) these files can become large rather quickly. To help you maintain your log files, the message logging service provides several options which enable you to more effectively archive log files. These options include:

- the number of past log files to keep
- the size in kilobytes of each log file
- stopping logging events or logging to a new file once the specified log file size is reached

Setting the Number of Event Logs to Keep

You can set the number of past log files to keep with the option **-k** *integer*. When not specified, the default number of log files to keep is one.

Setting the Event Log Size

The size the log file reaches is determined by the option **-p** *integer*, where the *integer* is the maximum size in kilobytes the log file is allowed to reach before logging is halted or written to another file.

Setting Error Log Behavior

Once the maximum log file size is reached, you can use the **-b** option to specify that event logging either **halt** or **rollover**. When not specified, the default behavior is for logging to roll over.

If you specify that the log file roll over, the file is renamed by giving it the suffix **.1**. If past log files exist, each is renamed by incrementing its file extension by one so that ***filename*.1** is always the most recent log file.

> **Note:** If you choose to halt the message logging service once the maximum log file size is reached, you can set an alarm to warn you when the log file is near capacity. Set the alarm with the **-a** *integer* option, where *integer* is the size in kilobytes the log file should reach when the alarm is recorded. By default, there is no alarm.

Assigning a Private Logging Service

You can log events for specific services by assigning a private logging service to a specific service (also called a *producer*). To assign an instance of **mnlogsrv** to a specific producer, include the **-c *producer*** option. Producers are specified with the format:

host:PID

where:

host the server's hostname

PID the process ID of the producer (process) you wish to assign to **mnlogsrv**

For example, **oracle-sun:1725** specifies the service with process ID 1725 running on the server oracle-sun.

Example 5–3 Assigning a Private Logging Service

```
% mnlogsrv -c oracle-sun:1225 -f "maxsev 6" -m $ORACLE_HOME/vs30/mesg -p 1024
-o $ORACLE_HOME/mn33/log/mzlog_1225 -t textfile
```

This example assigns a private logging service to process ID 1225 running on the server oracle-sun. Note that the log file specified with **-o** uses the process ID to identify it from the default log file.

Viewing Log Files

Oracle Video Server's log file can help you monitor and manage your server installation. The log file contains both information and error messages generated by *server events* (error or transaction messages generated by Oracle Video Server services and utilities). By viewing Oracle Video Server's log file, you can troubleshoot and correct possible problems.

You can view the Oracle Video Server log file either from the server computer's console, or you can monitor and manage using the VSM console.

Viewing Log Files with Oracle Video Server Manager

You can use Oracle Video Server Manager to monitor your Oracle Video Server system. Oracle Video Server Manager allows you to:

- view the status of critical Oracle Video Server services
- view the error log
- view server status

For information on using Oracle Video Server Manager to view log files, refer to:

- *Getting Started with Oracle Video Server Manager*
- Oracle Video Server Manager online Help

Viewing Log Files From the Server Console

To view log files from the server computer's console, use **mnlogreader**, the log reader utility. To view a log file with **mnlogreader**, you must specify:

- a filter level through which to view the log file. Specify the filter level with the **-f *filter*** option.
- the name and location of the log file you wish to view. Specify the log file name and path with the **-i *filename*** option.
- the log file type (binfile or textfile). Specify the log file type with the **-t *filetype*** option.

Example 5–4 Using mnlogreader to view a log file

```
% mnlogreader -f "maxsev 6" -i $ORACLE_HOME/mn33/log/mzlog -t textfile
```

This example specifies the log file **$ORACLE_HOME/mn33/log/mzlog**, a **textfile** log file. The filter level specified is "**maxsev 6**" which returns all logged events. The following sections discuss **mnlogreader's** viewing options.

Setting the Filter Level

mnlogreader allows you to specify the level of events you view using filters. You can configure **mnlogreader** filters to display either all log messages up to a specified severity level (**maxsev**), or only those log messages of a certain severity level (**sev**). The severity levels are described in Table 5–1.

To specify that **mnlogreader** return all events where the severity level is less than or equal to the specified severity, specify "**maxsev** *integer*," where *integer* is one of the severity levels listed in Table 5–1. For example, to view all events contained by the log file, specify "**maxsev 6**." For example:

```
% mnlogreader -f "maxsev 6" -i $ORACLE_HOME/mn33/log/mzlog -t textfile -c
```

You can also specify that **mnlogreader** only return events of a specific severity level. To view events of a specific severity level, use the syntax "**sev** *integer*," where integer is one of the severity levels listed in Table 5–3. For example, to read events of only severity level 2, specify "**sev 2**." For example:

```
% mnlogreader -f "sev 2" -i $ORACLE_HOME/mn33/log/mzlog -t textfile -c
```

Log File Viewing Options

mnlogreader allows you to either show all of a log file in a continuous mode, or to view events as they are delivered to **mnlogsrv**, continuously, without looking at the log file. Viewing events as they are delivered to **mnlogsrv** is similar to using the **tail** utility to view a log file.

To view the log file continuously, specify the **-c** option when starting **mnlogreader**:

```
% mnlogreader -f "maxsev 6" -i $ORACLE_HOME/mn33/log/mzlog -t textfile -c
```

To view events as the are delivered to **mnlogsrv**, specify the **-e** option when starting **mnlogreader**:

```
% mnlogreader -f "maxsev 6" -e
```

File Types

To view a log file, you must specify the log file type you wish to view. Log file type is specified with the option **-t {textfile | binfile**} of **mnlogsrv**. For example, to specify that **mnlogreader** view ASCII text log files, specify **-t textfile**. When not specified the default is **binfile**. To find out what type of log file **mnlogsrv** is outputting, examine the Oracle Media Net configuration file (**mnstart**). View the **mnlogsrv** entry and examine the **-t** option to see what the file type is set to.

Example 5–5 Specifying log file types

```
% mnlogreader -f "maxsev 5" -i $ORACLE_HOME/mn33/log/mzlog -t textfile -c
```

This example specifies an ASCII text log file type. Note that **mnlogreader** only views binfile or textfile log files. If you choose to log event and error messages to the server's **syslog** service or console these options will record event and error messages as defined by your server operating system's configuration.

Using the mnlog Utility

You can run the **mnlogreader** utility by executing the file **$ORACLE_HOME/bin/mnlog**. To use the **mnlog** file:

1. Edit the **mnlogreader** entry contained in the file to view the log file for your system.
2. Type **mnlog** at the command prompt:

   ```
   % mnlog
   ```

6

Monitoring Clients

Oracle Video Server provides two administrative interfaces from which you can monitor client activity: Oracle Video Server Manager (VSM) and the UNIX command line. This chapter describes the different types of information you can obtain from each interface, as well as how to disconnect an active client from the server. Before continuing in this chapter, you should be familiar with how Oracle Video Server communicates with clients. To learn about this, refer to *Introducing Oracle Video Server*.

Topics in this chapter are:

Using Oracle Video Server Manager

Using Oracle Video Server Manager (VSM) you can monitor:

- program titles clients are playing.
- length of currently playing programs in hours, minutes, and seconds
- current position of the playing program in hours, minutes, and seconds
- information on logical content titles such as the number of clips the title contains.
- client viewing status: currently playing a stream, paused, or finished
- client's host name or network address. The network address is displayed if the host name is not available.
- transmission rate at which the video pump is streaming data to the client.

To learn about Oracle Video Server Manager, and its use in monitoring client usage, refer to:

- *Oracle Video Server Manager Getting Started Guide*
- Oracle Video Server Manager online Help

Using the UNIX Command Line

Oracle Video Server provides two command line utilities for client management:

- **vscsmdir**

  The session and circuit list utility (**vscsmdir**) provides information about active client sessions and the resources allocated to each session. Use **vscsmdir** to obtain:

  - client's device ID
  - client's network address
  - information about the communication channels between Oracle Video Server and the clients

- **vscsmkill**

  The session kill utility (**vscsmkill**) disconnects clients from Oracle Video Server.

Obtaining a Client's ID

Use **vscsmdir** with the **-a** option to obtain a list of active client connections. The boldface text highlights the ***client-id*** portion of the output. Note that the format of the ID is specific to the client. This example uses Oracle Video Client, but a client from another vendor may produce different output for the client ID portion of the output.

```
% vscsmdir -a
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ] \
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
ovc:oracle2-pc-418 27-Jan-1998 09:17:10.011 PST 2/0 [dupCD-P <1.0.41.667> ] \
[d-p-DIP+ (d-p--IP UDP:127.0.0.21:5101 3100000 bps)]
```

Other information listed by the **-a** option includes:

- client's physical address
- maximum bit rate a client's video pump will provide over the communication channel
- communication channels between Oracle Video Server and the client
- resources allocated on behalf of a client by Oracle Video Server

Refer to **vscsmdir** in Chapter 14, "Session-and-Circuit Service Utilities" to learn more about the output provided by the **-a** option.

Obtaining Maximum Bit Rate of a Channel

You can list the maximum bit rate the video pump will provide over a given channel with the **-b** option. Depending on the number of active clients, you may wish to specify a specific client with the **-i *client-id*** option. This example lists the maximum bit rate of the client identified as **ovc:oracle-pc-389**.

```
% vscsmdir -i ovc:oracle-pc-389 -b
ovc:oracle-pc-389 [] [(3100000 bps)]
```

Disconnecting a Client

If you need to shut down Oracle Video Server, you may want to first disconnect any clients connecting to the server. To disconnect a client:

1. Obtain a listing of all active clients with **vscsmdir**. The **-a** option lists all available client information, including the ***client-id***. This example highlights the client ID in boldface text:

```
% vscsmdir -a
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.8.666> ] \
[d-p-DIP- (d-p--IP UDP:144.25.117.208:1099 3100000 bps)
ovc:oracle2-pc-418 27-Jan-1998 09:17:10.011 PST 2/0 [dupCD-P <1.0.41.667> ]\
[d-p-DIP+ (d-p--IP UDP:127.0.0.21:5101 3100000 bps)]
```

2. Disconnect the client with **vscsmkill** by specifying the ***client-id*** as shown:

```
% vscsmkill ovc:oracle-pc-389
```

For Further Information

For further information on the Session-and-Circuit Service, refer to the following documents and sections:

- To learn about how Oracle Video Server communicates with clients, refer to *Introducing Oracle Video Server*
- To learn more about the **vscsmdir** and **vscsmkill** utilities, refer to Chapter 14, "Session-and-Circuit Service Utilities."

7

Oracle Media Data Store Tasks and Procedures

The Oracle Media Data Store (MDS) is an optimized file system designed to store multimedia data and deliver it in real time to multiple clients. Access to the MDS is controlled by **mdsdirsrv** (MDS directory server) and the MDS configuration is defined by the **voltab** configuration file.

For overview information about the MDS and how it works with the other components of Oracle Video Server, refer to *Introducing Oracle Video Server*.

Topics covered in this chapter are:

About the Oracle Media Data Store

The Oracle Media Data Store (MDS) is an optimized filesystem designed to store multimedia data and deliver it in real time to multiple clients. Access to the MDS is controlled by **mdsdirsrv** (MDS directory server) and the MDS configuration is defined by the **voltab** configuration file.

For information about the MDS and how it works, refer to *Introducing Oracle Video Server*. For information on the **mdsdirsrv** server process or the **voltab** configuration file, refer to Part II of this book, *Command Reference*.

Initializing the Oracle Media Data Store

The Media Data Store must be initialized upon installation of the Oracle Video Server software. Initialization writes the volume specification information from the **voltab** file to its disks. Oracle Video Server does not automatically initialize the MDS. This must be done using the procedures in this section.

To initialize an MDS volume, do the following:

- Make sure your **voltab** file contains the correct volume specifications
- Run **mdsvolinit** to initialize the volume.
- Run **mdsvolstat** to determine the correct maximum bandwidth (**maxbw**) for the volume

About the Voltab File

The **voltab** file, which defines the MDS configuration, is installed in **/$ORACLE_HOME/vs30/admin**. This file initially contains volume specification information supplied at installation. This volume is a “non-real-time” MDS volume, so called because, regardless of the load placed on the MDS, the volume does not deny a request from any client. This mode, while efficient in loading content into the MDS, can result in lower video quality to client devices. The following example **voltab** specification is a “non-real-time” volume specification because it does not contain a maximum bandwidth, or **maxbw** value:

```
video striped,width=32,raidsize=4 spares=/dev/rdsk/c3t3d0s6 /dev/rdsk/c{0-3}t{0-3}d0s6
```

Note that you should take care to avoid leading spaces before the name of the volume, or extra spaces between the volume parameters.

For more information on **voltab,** see **“voltab” on page 11-5**. For more information on considerations for creating volume entries in the **voltab** file, see Chapter 2, “System Planning for the Oracle Video Server”.

Using mdsvolinit to Initialize the MDS

Use the **mdsvolinit** utility to write the definition for each MDS volume from the **voltab** file to its disks. Perform this step only:

- before you start **mdsdirsrv** for the first time after installing the OVS software
- if you subsequently modify an MDS volume definition in the **voltab** file and need to re-initialize the volume before restarting **mdsdirsrv**
- if you want to erase all files from a volume

To initialize an MDS volume, you must first start Oracle Media Net (OMN). OMN enables communication among the OVS components so that the **mdsvolinit** utility can write information to the MDS volume.

To start Oracle Media Net alone, initialize a volume, then stop OMN, perform these steps:

1. Log in as the **oracle** software owner and change to the directory **$ORACLE_HOME/mn33/admin.**

   ```
   % cd $ORACLE_HOME/mn33/admin
   ```

2. Execute the OMN startup script, **mnstart**:

   ```
   % ./mnstart
   ```

3. Initialize the MDS volume using the **mdsvolinit** utility with the **-s** option to initialize the volume specification; the **-t** option to initialize the table of contents for the volume; and the **-f** option to specify the complete path to the voltab file. Finally, specify the name of the volume as shown below. This example initializes the MDS volume **video**:

   ```
   % mdsvolinit -s -t -f $ORACLE_HOME/vs30/admin/voltab video
   ```

This command writes information from the voltab file to the MDS disks. This process allows the media data store directory server (**mdsdirsrv**) to access the disks as an MDS volume set.

CAUTION: Using -t, which initializes a volume, erases all content from the specified volume. Once an MDS volume has been initialized, its properties (i.e. disk configuration, stripe width, raid size) cannot be modified without destroying all the volume's content. If you wish to modify an MDS volume, first back up all of its content, so it can be reloaded into the modified volume.

For more information on options to the **mdsvolinit** command, refer to **mdsvolinit** on page 12-40.

4. Execute the script **mnstop**. This stops all OMN components.

```
% ./mnstop
```

For more information on starting and stopping Oracle Video Server and Oracle Media Net, see Chapter 4, "Starting and Stopping Oracle Video Server".

Using mdsvolstat to Determine Maximum Bandwidth

To create a real-time MDS volume, where access to the volume is limited to a predefined maximum to ensure quality of service, you must obtain and set the **maxbw** value. **maxbw** specifies the maximum rate at which all Oracle Video Server server processes and clients can together read and/or write to an MDS volume.

Follow these steps to obtain the correct **maxbw** value for your MDS volume and mount the MDS volume in real-time mode.

1. Change to the directory **/$ORACLE_HOME/mn33/admin** and start Oracle Media Net.

```
% cd /$ORACLE_HOME/mn33/admin
% ./mnstart
```

2. Use **mdsvolstat** with the **-a** option to determine the **maxbw** of the MDS volume. Use the **-f** option to locate the **voltab** file used to define MDS volume being queried, followed by the name of the MDS volume you wish to query. This example queries the volume **video2** for its **maxbw** value:

```
% mdsvolstat -a -f $ORACLE_HOME/vs30/admin/voltab video2
Recommended maxbw setting for volume video2 is: 55 Mbps
```

For more information see **mdsvolstat** on page 12-43

3. Edit the **voltab** file to include the **maxbw** value.

```
video2 striped,width=32k,raidsize=4,maxbw=55, dev/rdsk/c{4-7}t{0-3}d0s{6}
```

> **Note:** Take care not to introduce extra spaces into the volume specification when editing the **voltab** file.

4. Stop OMN:

```
% ./mnstop
```

5. You can now start Oracle Video Server with the **ovsstart** script using the modified **voltab** file (see step 1). The MDS volume will be mounted in real-time mode.Start Oracle Video Server with the **ovsstart** script, located in the directory **$ORACLE_HOME/vs30/admin**:

```
% cd $ORACLE_HOME/vs30/admin
% ./ovsstart
```

6. Verify that the MDS volume is mounted in real-time mode using the **mdsdir** utility with the **-v** option. This will display the total available bandwidth of the volume.

```
% mdsdir -v /mds/video2
Volume name:                    video2
Creation time:                  Jan 13 10:40:35
Free TOC entries:               507
Total bandwidth in Mbps:            55
  available                     55
  allocated:                    0
  RT denials:                   0
  Non-RT denials:               0
```

Administering the Media Data Store

Basic procedures and utilities necessary to administer the MDS are the following:

- Setting the MDS_CWD Environment Variable
- Assigning Read-Write Modes

- Using MDS Utilities
- Creating Additional MDS Volumes
- Changing the Name of an MDS Volume
- Defragmenting the Oracle Media Data Store
- Enabling FTP Access to the MDS

Setting the MDS_CWD Environment Variable

When you use Oracle Video Server utilities, you should enter a full pathname to refer to any MDS file. This is because the MDS requires a complete path to the volume that contains the file when you perform an operation. If you have only one volume in your MDS, or mainly reference a certain volume, you may find it easier to set the path in the **MDS_CWD** environment variable, and omit the volume path in the command itself. The **MDS_CWD** environment variable prepends MDS path information to a filename that is specified without a path.

Examples

These examples use **vstagprint** to print the tag information in the tag file **oracle1.mpi** from the MDS volume **video**.

First, use **mdsdir** to show all available MDS volumes and files on the server:

```
% mdsdir
Volume /mds/movies (rw): 2 matches
cover.mpg          cover.mpi
Volume /mds/video (rw): 2 matches
oracle1.mpg     oracle1.mpi
```

Example 1 This example uses **MDS_CWD** as the path name to find the tag file **oracle1.mpi**:

```
% setenv MDS_CWD /mds/video
% vstagprint oracle1.mpi
Tag file version: 5.0
Current code is version 5.0, back-compatible to version 5.0
…
```

Example 2 In this example, **MDS_CWD** is pointing to the wrong volume (named **movies**), therefore the tag file **oracle1.mpi** is not found:

```
% setenv MDS_CWD /mds/movies
% vstagprint oracle1.mpi
```

```
mkctPrintTagFile: cannot open tag file oracle1.mpi
error while attempting to print tag file
error occurred in vstagprint, exiting
```

Example 3 In this example, **MDS_CWD** is set to the volume **video**, but the user specifies the volume **movies**. If you provide an explicit MDS volume path it will override **MDS_CWD**:

```
% setenv MDS_CWD /mds/video
% vstagprint /mds/movies/oracle1.mpi
Tag file version: 6.1
Current code is version 6.1, back-compatible to version 6.1
…
```

Example 4 In this example, **MDS_CWD** is not set and the MDS path is not specified. This produces the following error message:

```
% unsetenv MDS_CWD
% vstagprint oracle1.mpi
mkctPrintTagFile: cannot open tag file oracle1.mpi
error while attempting to print tag file
error occurred in vstagprint, exiting
```

Assigning Read-Write Modes

You can mount MDS volumes in read-write mode or read-only mode.

Read-write mode

A volume mounted in read-write mode allows video server processes to both read and write to the volume. To perform administrative tasks involving an MDS volume—such as loading content—the volume must be mounted in read-write mode.

Read-Only Mode

A volume mounted in read-only mode allows video server processes to read but not write to an MDS volume.

Examples

The following examples show the syntax used by **mdsdirsrv** to mount MDS volumes using varying read-write modes.

Example 1 This example uses **-W** to mount all volumes listed in the **voltab** file in read-write mode:

```
% mdsdirsrv -W -f $ORACLE_HOME/vs30/admin/voltab
```

Example 2 This example mounts all volumes in read-write mode with **-W** except for the volume **video3,** specified with the **-l** option, which is mounted in read-only mode:

```
% mdsdirsrv -W -l video3 -f $ORACLE_HOME/vs30/admin/voltab
```

Example3 This example, mounts all volumes in read-only mode with **-L**, except for the volume **video,** specified with the **-w** option, which is mounted in read-write mode:

```
% mdsdirsrv -L -w video -f $ORACLE_HOME/vs30/admin/voltab
```

Using MDS Utilities

Loading and managing content in MDS should be handled using Video Server Manager (VSM). You must however, use the command line to invoke utilities for more complex tasks. Chapter 12, “Oracle Media Data Store (MDS) Utilities” contains a full listing of the utilities that perform file operations on the Media Data Store. Some of the basic utilities are mentioned here for basic orientation to listing, copying and removing MDS files.

mdsdir

mdsdir provides information about an MDS volume and the files it contains. Use **mdsdir** with the **-a** option to list all files, including those which have been removed but not overwritten. Use **mdsdir** with the **-l** option to list all file names, sizes, dates and lock status. For example:

The command **mdsdir -a**:

```
% mdsdir -a /mds/video
Volume:  /mds/video (ro) 7 matches
oracle1.mpg  oracle4.mpg
oracle1.mpi  oracle4.mpi
oracle2.mpg  oracle5.mpg
oracle3.mpg
```

The command **mdsdir -l**:

```
% mdsdir -l /mds/video
Volume:  /mds/video 7 matches
```

```
 294m  Feb 26 17:09 rw oracle1.mpg
 273b  Feb 26 17:19 rw oracle1.mpi
  19m  Feb 26 16:57 rw oracle2.mpg
  94m  Feb 26 16:57 r- oracle3.mpg
  10m  Feb 26 16:57 rw oracle4.mpg
 172b  Mar 18 00:04 r- oracle4.mpi
  10m  Feb 26 16:57 rw oracle5.mpg
```

See **"mdsdir" on page 12-14** for a more extensive listing of the **mdsdir** command.

mdscopy

mdscopy copies one file to another or a number of files to an MDS volume or to a directory on the host. To copy a single file from one volume to another, you must use the **mdscopy** command and specify the complete path to the volume. For example:

```
% mdscopy /mds/video/oracle1.mpg /mds/video2/oracle1.mpg
```

To copy multiple files from one volume to another, use **mdscopy** and use a wildcard (*). For example:

```
% mdscopy "/mds/video/*" /mds/video2
```

See **mdscopy** on page 12-7for a more extensive listing of the options of **mdscopy**.

mdsdelete

To delete a file from the MDS, use the **mdsdelete** command.

```
% mdsdelete /mds/video/oracle1.mpg
```

To delete multiple files use the **mdsdelete** command and the wildcard (*). For example:

```
% mdsdelete "/mds/video/*.mpg"
```

See **mdsdelete** on page 12-12for more information and specific options to the **mdsdelete** utility.

Creating Additional MDS Volumes

Invariably, you will need to add additional storage capacity to Oracle Video Server as your video library grows. Since you cannot add disks to an MDS volume without erasing the data it contains, you will need to create additional volumes. Follow these steps to add volumes to an existing MDS system:

1. Determine the amount of storage capacity and bandwidth requirements for the volume you plan to create. For information on this, refer to Chapter 2, "System Planning for the Oracle Video Server."

2. Once you have determined the storage and bandwidth requirements, and have assembled the MDS disk system, stop the OVS with the **ovsstop** script located in **$ORACLE_HOME/vs30/admin**:

```
% cd $ORACLE_HOME/vs30/admin
% ./ovsstop
```

3. Edit the **voltab** file to include the new volume definition. The example below adds the volume **video2** to the **voltab** file:.

> **CAUTION :** The following example shows a voltab entry divided across two lines. When editing a voltab entry it should all be on one line. The only division of lines should occur when your text editor wraps from one line to the next. Do not place a hard return in a voltab entry**.**

```
video striped,width=32k,raidsize=4,maxbw=55, dev/rdsk/c{0-3}t{0-3}d0s{6}
video2 striped,width=32k,raidsize=4, dev/rdsk/c{4-7}t{0-3}d0s{6}
```

For information on the **voltab** file and its syntax, refer to **voltab** on page 11-5.

4. Start Oracle Media Net with the **mnstart** script located in **$ORACLE_HOME/mn33/admin**:

```
% cd $ORACLE_HOME/mn33/admin
% ./mnstart
```

Oracle Media Net allows communication to take place among Oracle Video Server components so that **mdsvolinit** can write to the new volume.

5. Initialize the new volume using **mdsvolinit** with the **-s -t** and **-f** flags. The example below initializes the volume **video2**:

```
% mdsvolinit -s -t -f $ORACLE_HOME/vs30/admin/voltab video2
```

mdsvolinit writes information from the **voltab** file to the disks used by the volume **video2**. Note that the **-s** and **-t** options used in this example should only be used to initialize a new MDS volume. Using this combination of options erases all data contained on the disks.

6. Stop Oracle Media Net with the script **mnstop** located in **$ORACLE_HOME/mn33/admin**.

```
% ./mnstop
```

7. Start the OVS with the **ovsstart** script located in **$ORACLE_HOME/vs30/admin**.

```
% cd $ORACLE_HOME/vs30/admin
% ./ovsstart
```

8. Use **mdsdir** with the **-f** option to verify that you can contact the new volume. This lists the free space available in the volume:

```
% mdsdir -f /mds/video2
Volume /mds/video2 (rw): 1 match
2400 MB's of free space in 1 fragment(s)
largest free block is 2400 MB's
```

The volume created is a non-real-time volume since no **maxbw** (maximum bandwidth) value is specified in the **voltab** entry. You may wish to load video content into the new volume at this time. Loading content into a non-real-time volume gives more bandwidth to the utilities used to copy and tag video content. To create a real-time volume, which is designed to stream video and audio content with real-time constraints, you need to run **mdsvolstat -a**, to determine the **maxbw**. The value is then added to the **voltab** specification for the volume, establishing a maximum bandwidth, which acts to assure highest quality video.

See "Using mdsvolinit to Initialize the MDS" on page 7-3 for more information.

Changing the Name of an MDS Volume

To change the name of an MDS volume, without destroying its contents, you must re-initialize the volume's specification with the **mdsvolinit** utility and retag all content with **vstag**. This example changes the MDS volume name from **video** (as shown below) to **volume1**:

1. The current volume, as shown in the **voltab** entry below, is **video**.

```
video maxbw=55,striped,width=32k,raidsize=4, dev/rdsk/c{0-3}t{0-3}d0s{6}
```

2. Shut down the OVS using the **ovsstop** script located in **$ORACLE_HOME/vs30/admin**:

```
% ./ovsstop
```

3. Start OMN with the **mnstart** script located in **$ORACLE_HOME/mn33/admin**:

```
% cd $ORACLE_HOME/mn33/admin
```

```
% ./mnstart
```

4. Make a backup copy of the **voltab** file.
5. Edit the **voltab** entry and replace the name **video** with **volume1** as shown below:

   ```
   volume1 maxbw=55,striped,width=32k,raidsize=4, dev/rdsk/c{0-3}t{0-3}d0s{6}
   ```

6. Use **mdsvolinit** with the **-s** and **-f** options to rename the MDS volume.

   CAUTION: Do not use the -t option, which will overwrite the volume's table of contents and erase all content.

   ```
   % mdsvolinit -s -f $ORACLE_HOME/vs30/admin/voltab volume1
   Do you really want to change attributes of volume 'volume1'? (yes, no)
   yes
   Attributes of volume 'volume1' recorded.
   ```

7. Shut down OMN with the **mnstop** script located in **$ORACLE_HOME/mn33/admin**:

   ```
   % cd $ORACLE_HOME/mn33/admin
   % ./mnstop
   ```

8. Start the OVS with the **ovsstart** script located in **$ORACLE_HOME/vs30/admin**:

   ```
   % cd $ORACLE_HOME/vs30/admin
   % ./ovsstart
   ```

9. Use **vstagpatch** to modify the tag files to point to the new location of the MPEG files. This will save time if the content files are very large.

   ```
   % vstagpatch -c -n /mds/volume1/oracle1.mpg /mds/volume1/oracle1.mpi
   ```

Defragmenting the Oracle Media Data Store

Performing file operations on the MDS, especially removing and truncating files, can leave gaps between files in the MDS volume. Although the MDS directory server (**mdsdirsrv**) stores other files in these gaps if they fit, this fragmentation results in wasted disk space. Because the performance of the MDS is not affected by disk fragmentation, defragmenting the MDS only reduces potentially wasted disk space; it does not improve performance.

The **mdsdefrag** utility can be used whenever the system is on-line—while the MDS is streaming video or while content is being added to or deleted from an the MDS volume.

> **Note:** The **mdsdefrag** utility is selective when moving files. If the amount of data to be relocated is very large in relation to the perceived benefit, then **mdsdefrag** might not defragment the volume completely. In most cases, it will completely defragment a volume.

The defragger will continually strive to reduce the amount of fragmentation in an MDS volume, consolidating free space. It may not however, be able to defragment a volume completely such that all free space is collected into one contiguous space.

Example This example defragments the MDS volume **video**:

```
% mdsdefrag video &
```

Enabling FTP Access to the MDS

The MDS FTP server (**mdsftpsrv**) allows any FTP client—regardless of platform—to access files contained in the MDS.

Features and Limitations of MDS FTP

- MDS FTP does not provide access to host files. To access host files, use the existing FTP server provided by the operating system or networking vendor. The MDS FTP server co-exists with the standard FTP server by using a different port number.
- Since the data stored on MDS filesystems is digitized video and audio data, MDS FTP provides support only for binary file transfers. ASCII data transfer is not supported.
- **mdsftpsrv** can transfer data to either an FTP client or another FTP server (either **mdsftpsrv** or **ftpd**) using passive mode.
- You can use remote management commands to list, rename, and delete MDS files.
- MDS FTP can serve multiple concurrent connections.

Examples

Example 1 This example starts **mdsftpsrv** with the **-g** option, restricting access to users belonging to the same group as the owner of the **mdsftpsrv** process. The port specified with **-p** is 1621 as recommended by Oracle. Note that **mdsftpsrv** must listen on its own port so that it can co-exist with other FTP daemons:

```
% mdsftpsrv -g -p 1621 &
```

Example 2 This example starts **mdsftpsrv** with **-w**, giving access to any authorized user on the system:

```
% mdsftpsrv -w -p 1621 &
```

Example 3 In this example, **mdsftpsrv** is started on a system port, which is any port numbered 0 to 1023. To start **mdsftpsrv** on a system port, you must be logged in as **root**. This example specifies that **mdsftpsrv** listen on system port 621 with the **-p** option, while restricting access to those users belonging to the same system group with **-s**:

```
% mdsftpsrv -s -p 621 &
```

Example 4 **mdsftpsrv** provides for additional security through the use of an encrypted password. You must first obtain the encrypted version of your UNIX account password:

ypmatch *username* passwd
where:

username your UNIX login ID.

For example, to obtain the encrypted password for user **sflyte**, enter **ypmatch sflyte passwd** at your shell prompt:

```
% ypmatch sflyte passwd
sflyte:iMMElWbxWACVk:20211:520:Sebastian Flyte,,,:/home/sflyte:/bin/csh
```

The second field of the returned output contains the encrypted password for user **sflyte**; in this instance **iMMElWbxWACVk**.

To specify the encrypted password as the MDS FTP login password, use the **-R** option to specify the **mds.alternate** resource. The syntax for the resource statement is:

mds.alternate=*password*

where:

password is the encrypted output of **ypmatch**.

To use the encrypted password, use the **-a** option in conjunction with **-R** and the **mds.alternate** resource descriptor and password:

```
% mdsftpsrv -a -R mds.alternate=iMMElWbxWACVk
```

> **Note:** The format of the encrypted password is specific to your operating system.

Example 5 This example starts **mdsftpsrv** with **-A**, specifying a default allo size of 12 Megabytes:

```
mdsftpsrv -A 12000000
```

The **-A** option sets a default file creation size which is the maximum size of an MDS file FTP will create within the MDS. The **-A** option assumes that the **allo** command has not been specified by the FTP client. When using **mdsftpsrv** with **-A**, the default allo size should be set high enough to accommodate any anticipated file transfer. Sending the **allo** command for individual file transfers with the FTP client therefore becomes unnecessary. If you attempt to FTP a file whose size exceeds that set with **-A**, the file transfer will fail.

You may set the value at the command line when -A is not set, or as an override to the setting, using the **allo** command as discussed below in "Sending the Allo Command".

Copying Content Using Passive FTP

A discussion and examples of using passive FTP is found in the section “Copying Content Using Passive FTP” on pages 4-10 through 4-11 in the *Oracle Video Server Content Administrator’s Guide.*

Sending the Allo Command

When you FTP files into the MDS you must provide the MDS file system with the maximum size estimation of the file you wish to transfer. Use the **allo** command to allocate space for the file. For example:

Example 7–1 allo command in UNIX ftp

```
ftp>quote allo n
where:
    n is the number, in bytes, of the maximum size of the file.
```

Example 7–2 allo command for DOS

```
literal allo n
```

You may alternatively specify the **-A** option to **mdsftpsrv** which sets the maximum file size for any file created within the MDS. Of course, if you exceed the maximum file size you set using **-A**, the process will fail.

Further Reading

For more information on **mdsftpsrv**, refer to **mdsftpsrv** on page 11-13.

Repairing the Media Data Store

By checking Oracle Video Server's error logs, as described in Chapter 5 "Monitoring and Maintaining Log Files," you might learn that a hard disk has failed. The following procedures are covered in this section:

- Disk Hot-Swapping
- Disk Sparing
- Substituting the Spare Disk for a Failed Disk
- Replacing the Failed Disk

Disk Failure

If your system supports hot-swappable disks, follow instructions for disk hot-swapping; otherwise, use disk sparing procedures.

Disk Hot-Swapping

Disk hot-swapping enables you to replace a failed hard disk without bringing the system down.

Follow these steps to remove the disk and rebuild a new one:

1. Place the disk in rebuild mode using the **mdsdiskmode** utility with the **-r** option.

   ```
   mdsdiskmode -f $ORACLE_HOME/vs30/admin/voltab -r failed-disk
   ```

2. Remove the disk and replace it with a new one.
3. Rebuild the data to the disk with the **mdsrebuild** utility:

   ```
   % mdsrebuild -f $ORACLE_HOME/vs30/admin/voltab failed-disk
   ```

When the rebuilding process has finished, the disk automatically returns to normal mode.

Disk Sparing

Disk sparing allows you to replace a failed hard disk without removing the Oracle Video Server from service. To use disk sparing you must allow for a spare disk during the MDS configuration process. The spare disk for a volume is specified in the **voltab** file. Refer to the section **"voltab" on page 11-5** for more information on creating a **voltab** file using a spare disk.

Substituting the Spare Disk for a Failed Disk Follow these steps to transfer the data from a failed disk to a spare:

1. Spare the broken disk using the **mdsdiskmode** utility with the **-s** option. This command redirects all read-write operations from the failed disk to the spare disk, and prepares the spare for rebuilding:

   ```
   % mdsdiskmode -f $ORACLE_HOME/vs30/admin/voltab -s spare-disk failed-disk
   ```

2. Rebuild the data from the failed disk to the spare disk with the **mdsrebuild** utility:

   ```
   % mdsrebuild -f $ORACLE_HOME/vs30/admin/voltab failed-disk
   ```

   Note that you specify the name of the failed disk when rebuilding data to the spare. Using **mdsdiskmode** causes the spare to be recognized in place of the failed disk by **mdsdirsrv**.

   The spare disk now takes the place of the failed disk until administrative downtime can be arranged to install a new disk.

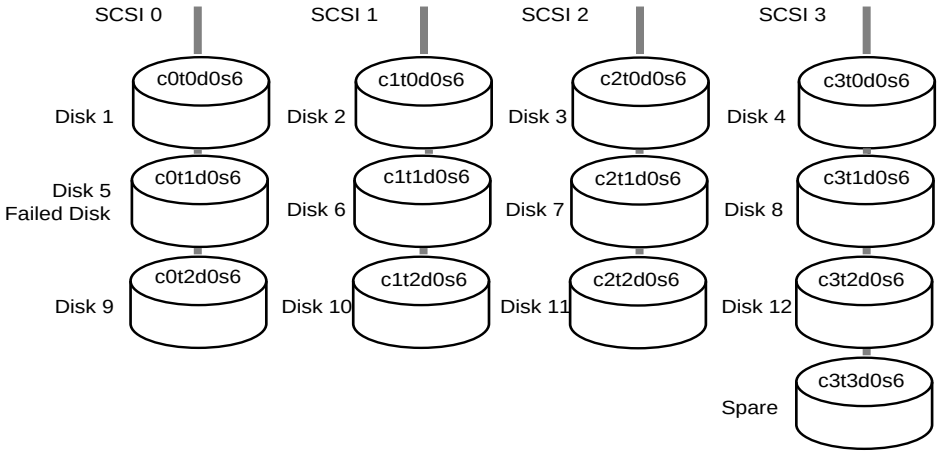
Example The following example shows how to use a spare disk in the event of a disk failure.

Assume the **voltab** file defines the volume **video** with 12 disks in 3 RAID sets of 4 disks each, and one spare.

> **Note:** The disk naming conventions used in this example may differ from your server platform's. Refer to your *Oracle Video Server Installation Guide* for disk naming conventions for your server platform.

```
video striped,width=32k,raidsize=4,maxbw=55,spares=/dev/rdsk/c3t3d0s6 /dev/rdsk/c{0-3}t{0-2}d0s6
```

Figure 7–1 MDS volume video



Suppose you receive a message that the second disk in the first SCSI chain has failed. This disk is identified as **/dev/rdsk/c0t1d0s6** (Disk 5). The spare disk is identified as **/dev/rdsk/c3t3d0s6** (spare). This disk is on the fourth SCSI chain of the volume **video**.

To enable the spare disk and rebuild data from the remaining disks to the spare, follow these steps:

1. Spare the failed disk **/dev/rdsk/c0t1d0s06** (Disk 5) using **mdsdiskmode** with the **-s** option, which redirects all read-write operations to the spare disk **dev/rdsk/c3t3d0s6**, and prepares the spare for rebuilding.

   ```
   % mdsdiskmode -f $ORACLE_HOME/vs30/admin/voltab\/dev/rdsk/c3t3d0s6 \
   /dev/rdsk/c0t1d0s6
   ```

2. Rebuild the data from the failed disk **/dev/rdsk/c0t1d0s06** (Disk 5) to the spare disk **dev/rdsk/c3t3d0s6**, with **mdsrebuild**:

   ```
   % mdsrebuild -f $ORACLE_HOME/vs30/admin/voltab /dev/rdsk/c0t1d0s6 \
   /dev/rdsk/c3t3d0s6
   ```

   mdsrebuild reads the remaining disks in the RAID set to obtain their parity information. This information is used to reconstruct the data stored on the failed disk and write it to the spare.

Replacing the Failed Disk

When administrative downtime can be arranged, swap the failed disk with the rebuilt spare and replace the spare disk with a new unit.

1. Shut down the server.
2. Physically replace the failed disk (Disk 5) with the spare disk. Replace the previous spare disk with a new unit so the volume can withstand another disk failure.
3. When you bring the server back up, start OMN. This will allow you to unspare the failed disk.

   ```
   % ./omnstart
   ```

4. Unspare the failed disk **/dev/rdsk/c0t1d0s6** (Disk 5) with the **mdsdiskmode** utility using the **-u** option:

   ```
   % mdsdiskmode -f $ORACLE_HOME/vs30/admin/voltab -u /dev/rdsk/c0t1d0s6
   ```

   You must unspare a disk so that **mdsdirsrv** reads and writes to the correct disk, which is **/dev/rdsk/c0t1d0s6** (Disk 5) and not the replacement spare disk **/dev/rdsk/c3t3d0s6**.

   WARNING: Make sure to unspare the disk before starting mdsdirsrv. If you do not unspare the disk, it will not be recognized by mdsdirsrv as a usable disk.

5. Stop OMN with the **mnstop** script.

   ```
   % ./mnstop
   ```

   Oracle Video Server is now available for normal operation. Start Oracle Video Server with the **ovsstart** script.

   ```
   % ./ovsstart
   ```

Archiving MDS Content

As you continue to add files to the MDS, you may find that you wish to off-load content which is in low demand in order to make space for more content files. You may use the **mdstar** or the **mdsftp** utilities to transfer files to backup storage. If you have use of a hierarchical storage device, you may use **mdshmctl** to transfer content to a tertiary storage device. Each method is discussed below.

Archiving Files Using mdstar

Files can be transferred between the MDS and a tertiary storage device, or a compressed tar file, using the **mdstar** utility.

Example 1 This example archives the file **/mds/volume1/oracle.mpg** to the device **/dev/rmt/0**.

```
% mdstar -c -T -f /dev/rmt/0 /mds/volume1/oracle.mpg
a oracle.mpg, 1610612789 bytes, 1 segment(s), 293 kb/s
```

This example uses the default blocking factor of 128. To ensure that you are using the correct blocking factor, refer to your server platform documentation.

Note that this example and others in this section use the **-T** option. This option enables verbose mode, which lists additional information about actions taken on files.

Example 2 This example archives all files in **/mds/volume1** with the extension **.mpg** to a file:

```
% mdstar -c -T -f /tmp/film.tar "/mds/volume1/*.mpg"
a star_wars.mpg, 1610612789 bytes, 1 segment(s), 278 kb/s
a clip1.mpg, 3610612 bytes, 1 segment(s), 289 kb/s
```

Example 3 This example lists the contents of the archive device **/dev/rmt/0m**:

```
% mdstar -t -f /dev/rmt/0
oracle1.mpg, 1610612789 bytes, 3145729 blocks, 1 segments, Feb 12
11:18:26
oracle2.mpg, 3610612 bytes, 7052 blocks, 1 segments, Feb 12 11:19:05
```

Example 5 This example extracts files from the archive **film.tar** and places them in the MDS volume **volume1**.

```
% mdstar -x -f /tmp/film.tar -p /mds/volume1
```

Segmenting Large Files

If your host file system does not support files of 2 GB or greater, you can create files using *__integer1_integer2_file.ext* naming format, put them on tape, and **mdstar** will concatenate them into one large file named *file.ext* when you load them into a MDS volume.

To load a file larger than the supported capacity of the host file system, specify a limit size value with **-l**.

If you load a file larger than the limit size specified with **-l**, **mdstar** divides the file into multiple file segments, each no larger than the limit size, and loads each separately. Names of segment files have this form:

__integer1_integer2_file
where:

integer1 is a hexadecimal integer representing the byte position in the original file where this file begins.

integer2 is hexadecimal integer representing the size of the original file in bytes.

filename is the name of the original file.

For example, **mdstar** divides a 5 GB file named **oracle.mpg2** into these 3 files when the tape archive is created:

```
__0_140000000_oracle.mpg2
__7fffffff_140000000_oracle.mpg2
__ffffffff_140000000_oracle.mpg2
```

For more information refer to your UNIX tar documentation.

Archiving Files Using mdsftpsrv

In order to FTP files to or from an MDS volume, the MDSFTP server, **mdsftpsrv,** must be running**.** You may use most operating system FTP clients to transfer files across a network from a host file system to an MDS volume. **mdsftpsrv** does not provide access to host file systems. To access files on host file systems, use the FTP daemon provided by the operating system or network vendor. You must then use the **mdsftpsrv** specific port number, 1621, to enable the transfer.

For specific details on enabling FTP Access, see "Enabling FTP Access to the MDS" on page 7-13.

As the MDS contains only binary data in the form of digitized video and audio files, only the transfer of binary data is supported. ASCII transfer mode is not supported.

For specific examples and details on archiving MDS content files using FTP, see "Archiving MDS Content Files Using FTP" in the *Oracle Video Server Content Administrator's Guide*.

Archiving Using mdshsmsrv (Hierarchical Storage Management Service)

Oracle Media Data Store (MDS) files can reside in tertiary storage (tape or CD-ROM), on disk, or both. However, MDS client applications can access only those files that reside on disk.

Generally, disks serve as caches for files that are frequently accessed, or files that have been recently accessed. In contrast, tertiary storage devices serve as a backing store, providing much longer-term storage of MDS files. With a tertiary storage device, you can copy older (or less frequently accessed) MDS files to tape and remove the files from disk. Tapes are loaded into and unloaded from the storage device with a robotic arm or stacker. To manage the storage and transfer of these MDS files and tapes efficiently, the MDS system provides a hierarchical storage management (HSM) system.

HSM offers:

- file backup and restoration
- tertiary storage management utilities

Every MDS file that resides in tertiary storage has some associated metadata that describes its location in storage and how it should be handled by the HSM system. The metadata on an HSM file can be viewed using the **mdshsmdir** utility.

A files is registered, associating it with a particular tape or CD, and space is created for it in the HSM system using the **mdshsmctl** utility. The transfer of the file does not have to be done when the file is registered, although it can be done at the time using **mdshsmctl**. The status of the transfer of the file to tape can be verified using the **mdshsmdir** utility.

Once a file has been copied to tape or CD, it can be removed either by the **mdsdelete** utility.

The examples in this section are performed using the Ampex DST on a Sun Solaris platform.

> **Note:** To adapt these procedures for use with your particular storage device, refer to the platform-specific documentation.

Creating and Initializing the HSM Data Tables

HSM data files provide metadata information to the robotic arm or stacker used to load and unload tapes. HSM data tables serve as an index for the tapes and files registered in the HSM system.

Before initializing data tables, you must ensure that the HSM data files have been created in the Oracle Media Data Store (MDS). To create files, perform the following steps.

1. Set MDS_CWD to the appropriate volume name when creating the new files. For example, given the volume **movies**:

   ```
   % setenv MDS_CWD /mds/movies
   ```

2. Verify that you have initialized and granted the necessary read-write privileges for the appropriate MDS volume.

3. To initialize the HSM data table, perform the following steps:

   a. Ensure that the **mdsdirsrv** process is running.

   b. Use **mdshsmsrv** to create the new files:

      ```
      % mdshsmsrv -d /dev/amc0 -f 1000 -t 10 ampex movies
      ```

      The **-f** and **-t** options create tables capable of holding 1000 files and 10 tapes, respectively. The *control device name* for the stacker or robot is, in this case, **/dev/amc0**, while **movies** is the name of the MDS volume to be served by HSM. The *control device* is **ampex**.

      > **WARNING:** Once you have initialized these tables, do not run **mdshsmsrv** with the **-f or -t** options on these tables again. Doing so will overwrite all current information in the HSM database**.**

4. Ensure that the tables were successfully created by looking at the table-creation messages in the log file:

   ```
   { created new HSM tape datafile with 10 rows available }
   ```

```
{ created new HSM file datafile with 1000 rows available }
```

Starting the HSM Servers

To use the HSM system, start at least one single-data transfer server and one HSM server. If your HSM system uses multiple tape storage devices, consider starting additional data transfer servers. (Each data transfer server can manage only one tape storage device. If you want to use multiple tape devices at the same time, start one data transfer server per tape drive.)

To start the HSM server, perform the following steps:

1. Ensure that **mdsdirsrv** is running, and that the MDS volume is available for use.
2. Start the data transfer server and the HSM server using the commands:

   ```
   % mdsxfrsrv -r 5 -d 100 /dev/rdst0.1 &
   % mdshsmsrv -d /dev/amc0 ampex movies &
   ```

   This example starts the data transfer server for the tape device **/dev/rdst0.1** and the HSM server for the control device **/dev/amc0**, which will serve the volume **movies**.

   The log file contains the startup notices from both of these processes.

   The **-r 5** (retry) option indicates that failed loading operations should be retried five times before abandoning the operation. This is necessary for some vendors, which might not be able to immediately unload a tape when the file being transferred is closed. Retries occur at a fixed four-second interval.

   The **-d 100** (device internal address) option uses the Ampex-specific, internal address of the device (100) to invoke tape-manipulator commands. **The Ampex device requires that you specify this parameter.** Issue all other data-transfer commands via the primary device name (**/dev/rdst0.1**).

   Another important, optional parameter to the **mdsxfrsrv** process is the **-c** (cache) option. This flag enables you to set a larger memory cache when MDS is initialized. The default cache value works well for most systems, but if you find that copying takes longer than expected, increase this value to at least two times the largest tape block size of any tape. Refer to “Registering Tapes with the HSM” for more information about block size.

Registering Tapes with the HSM

After starting an HSM server, you must register each tape that will be loaded on the system. To register a tape, create a record in the HSM data table for each tape

using **mdshsmctl** with the **-ct** flag to create tapes. Use the **-v** flag to specify the volume. Tapes need only be registered once, when they are introduced to the system.

Example This example registers four tapes for the volume **movies**:

```
% mdshsmctl -ct -v movies -B lm -1 160k 000001
% mdshsmctl -ct -v movies -B lm -1 160k 000002
% mdshsmctl -ct -v movies -B lm -1 160k 000003
% mdshsmctl -ct -v movies -B lm -1 160k 000004
```

Note: This registration process only creates a *description* of the tape; it does not actually verify that such a tape is present on the system.

The **-B** option defines the tape's block size; all read or write operations to or from the tape will be made in blocks of this size. Pick a block size value that provides the optimal transfer speed.

The **-l** option specifies the approximate length of the tape, in "block-size" units. Be conservative when specifying the length of a tape; if you specify too long a length, you risk running out of room on the tape itself, resulting in a wasted transfer and additional system administration.

Note: Some tape devices may have special restrictions on block sizes. Be sure to check for these restrictions before specifying the block size.

Registering Files with the HSM

After you have registered a tape with the HSM system as described in the previous section, you can begin to register MDS files with HSM. Registering a file with the HSM associates the file with the registered tape. Before registering a file with a tape, ensure that:

- the file exists in MDS and has not already been registered with HSM
- there is sufficient space remaining on the registered tape

Note that no data is actually copied at this time; this process simply creates a binding between the MDS file and a registered HSM tape. The **-cf** flag, to create a file, is used with the **mdshsmctl** command.

Example This example registers the MDS file **oracle1.mpg** with the tape **000001**:

```
% mdshsmctl -cf oracle1.mpg 000001
```

Determining the HSM Directory of Files and Tapes

Once the **mdshsmsrv** process has been started, you can use the **mdshsmdir** utility to determine the state of HSM tapes, files, and their associated directories.

Example 1 This example lists all the files registered with the HSM:

```
% mdshsmdir
TapeID     Len(bytes) FNum  BlockNum FileName
------     ---------- ----  -------- --------
000001            256    1         1 /mds/movies/oracle1.mpg
000002            256    1         1 /mds/movies/oracle2.mpg
000003            256    1         1 /mds/movies/oracle3.mpg
000001            256    2         3 /mds/movies/oracle4.mpg
000002            256    2         3 /mds/movies/oracle5.mpg
```

where:

TapeID	is the name of the tape.
Len(bytes)	is the length (in bytes) of the file.
FNum	is the file-number position a file has on the tape.
BlockNum	is the first block number on the tape to which the file was written.
FileName	is the name of the file.

Example 2 This example lists all the tapes registered in the HSM:

```
% mdshsmdir -t
Volume Mount  BlockSz TotalBlks  UsedBlks  RsrvBlks FNum Plays  TapeID  Fmt
------ -----  ------- ---------  --------  -------- ---- ------ ------  ---
movies UNMNT     8192        10         0         4    0      0  000001 TAR
movies UNMNT     8192        10         0         4    0      0  000002 TAR
movies UNMNT     8192        10         0         2    0      0  000003 TAR
```

where:

Volume	is the name of the volume associated with the tape.

Mount	is the mount status of the tape. **MNT** indicates that the tape is mounted in a tape device. **UNMNT** indicates that the tape is not loaded in a tape device and is in the tertiary storage device.
BlockSz	is the block size (in bytes) defined for the tape.
TotalBlks	is the total number of blocks available on the tape.
UsedBlks	is the number of blocks that are currently being used to store data.
RsrvBlks	is the number of blocks that have been reserved for backup and copying files from the MDS volume.
Plays	is the number of times the tape has been restored or backed up.
Fmt	is the format of the files stored on the tape.

Example 3 This example lists all the files registered with the HSM, sorted by tape:

```
% mdshsmdir -s
TapeID     Len(bytes) FNum  BlockNum FileName
------     ---------- ----  -------- --------
000001            256    1         1 /mds/movies/oracle1.mpg
000001            256    2         3 /mds/movies/oracle4.mpg
000002            256    1         1 /mds/movies/oracle2.mpg
000002            256    2         3 /mds/movies/oracle5.mpg
000003            256    1         1 /mds/movies/oracle3.mpg
```

The **-s** option causes the output to be sorted by tape ID and position on the tape.

Copying/Backing Up Files to Tape

The file registration process described in "Registering Files with the HSM" does not actually copy data. To copy or backup a file, use **mdshsmctl** with the **-b** flag, indicating backup operation:

```
% mdshsmctl -b oracle1.mpg
```

where **oracle1.mpg** is the name of the file to be copied / backed-up.

> **Note :** The file must already be registered with HSM, as explained earlier.

Deleting Files from the MDS

Once a file has been copied to tape, you can remove the file image from disk whenever that disk space is required, by using **mdsdelete**.

Example 1 To delete the file **oracle1.mpg** from the MDS, use the command:

```
% mdsdelete oracle1.mpg
```

When the delete operation is completed, the associated disk space will be available in the MDS.

Deleting Files from the HSM

Example 2 To delete files from the HSM database, use the **mdshsmctl** utility with the **-d** and **-f** flag and supply the file to be deleted as the argument. This example deletes the file **oracle1.mpg**:

```
% mdshsmctl -df oracle1.mpg
```

Since HSM appends new files to the end of the tape, deleting files from the tape does not recover the associated space.

> **Note :** Before a tape can be deleted from the HSM database, you must first delete all the files which were on the tape from the database. This safety measure ensures that tapes containing active content are not deleted from the database.

Deleting Tapes from the HSM

When tapes need to be removed from the HSM database, so they can be replaced with newer or different tapes, use the **mdshsmctl** utility.

> **Note:** Before deleting a tape from the HSM database, you must first delete the database records for all the files which are on the tape. This safety measure ensures that tapes containing active content are not deleted from the database.

Example 1 To delete tape **000001** from the HSM, use the following command:

```
% mdshsmctl -dt 000001
```

Restoring Files from Tape

You can restore files from tertiary storage onto disk through the explicit use of the **mdshsmctl** utility.

Example 2 To restore the file **oracle1.mpg**, use the **mdshsmctl** with the **-r** flag, supplying the name of the file to be restored as the argument:

```
% mdshsmctl -r oracle1.mpg
```

Once the restore operation is completed, the file will be available in the MDS.

8

Using Real-time Video Feeds

Oracle Video Server's Real-time Feed Service allows you to capture and store content from a live video source, such as VHS, DVD, LaserDisc or video camera, in a single step. The content is also available for playback with a small amount of delay.

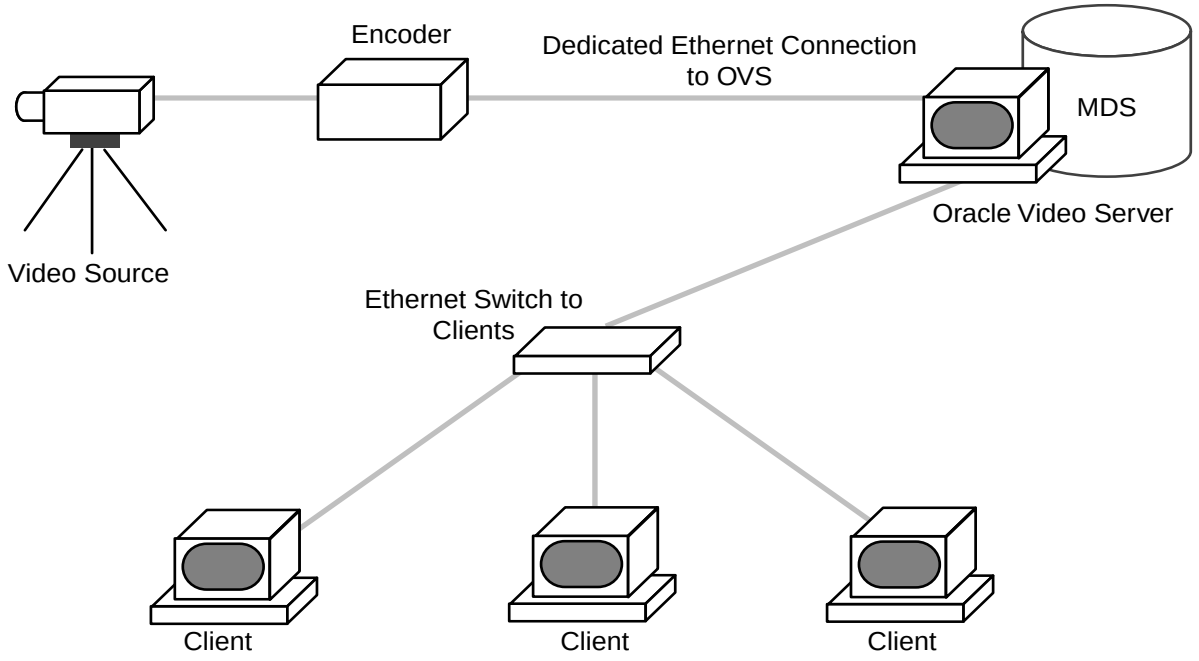
To enable the Real-time Feed service, you need to configure Oracle Video Server's **vsfeedsrv** process and a digital video encoder. To use the Real-time Feed service, you need:

- a live video source. This can be either a live broadcast or pre-recorded media on DVD, VHS, laser disc or video camera
- a VES-compliant digital video encoder
- Oracle Video Server
- Oracle Video Client
- a private, dedicated network connection between the encoder and the server running Oracle Video Server.
- two network connections on the video server, one for the encoder and one for the clients receiving the video feed.

The encoder digitizes the raw video and audio signal, compresses it and collects the metadata associated with the compressed stream. The Real-time Feed service, **vsfeedsrv**, processes the data stream and the metadata stream to produce two types of files within the Media Data Store: a tag file, containing meta data information for seeking and scanning, and compressed media files.

Figure 8–1 shows a Real-time Feed connection.

Figure 8–1 Real-Time Feed connection



Starting the Real-time Feed Session

Follow these steps to set up a real-time feed session:

- Connect the encoder and the Oracle Video Server
- Edit the ovsstart script to include **vsfeedsrv** in the startup sequence
- Configure and test the network connection
- Start a real-time feed session
- Shutting down the feed session

Connect the encoder and the Oracle Video Server

1. Connect the encoder to the Oracle Video Server using a private, dedicated Ethernet connection. The playback quality is directly affected by the network quality between the encoder and the Oracle Video Server.

Edit the ovsstart script to include vsfeedsrv in the startup sequence

2. Edit the file **$ORACLE_HOME/vs30/admin/ovsstart** to include **vsfeedsrv** in the startup sequence. Un-comment the following lines of the **ovsstart** script:

```
#   Option:
#   To use the Real-Time Feed option
#
echo -n "Starting Real-Time Feed service....."
$ORACLE_HOME/bin/vsfeedsrv &
sleep 5
```

3. Start Oracle Video Server.

   If OVS is already running, you may bring up **vsfeedsrv** by entering the following at the command line:

```
% vsfeedsrv &
```

   For more information on the **vsfeedsrv** command and its options, see "vsfeedsrv" on page 11-37.

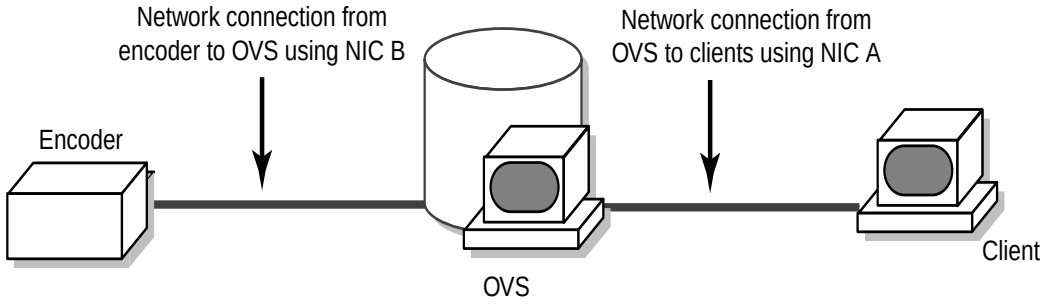
4. Ensure that **vsfeedsrv** is running by using the **mnorbls** utility on the Oracle Video Server to list all running services and their interfaces. If **IDL:ves/fcty:1.0** and **IDL:ves/sts:1.0** are active (status 1), proceed to the next section.

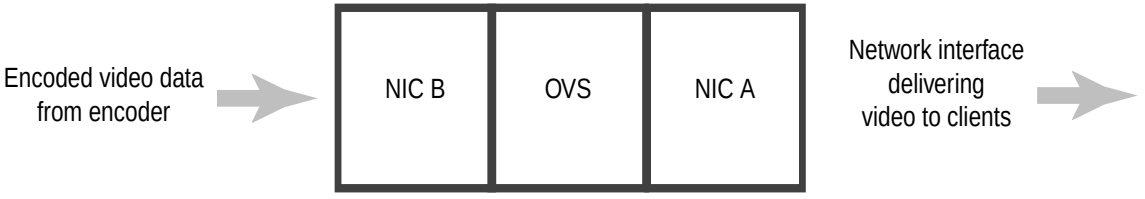
```
% mnorbls
INTERFACE                IMPLEMENTATION              HOST             ACTIVE
------------------------ --------------------------- ---------------------
...
IDL:mza/LgCtntFac:1.0    vscontsrv                   oracle-sun         1
IDL:mza/LgCtntMgmt:1.0   vscontsrv                   oracle-sun         1
IDL:mzc/cktMgr:1.0       mzcsrv                      oracle-sun         1
IDL:mzc/factory:1.0      mzcsrv                      oracle-sun         1
IDL:mzs/factory:1.0      vsstrmsrv                   oracle-sun         1
IDL:mzz/factory:1.0      mzzsrv                      oracle-sun         1
IDL:mzz/sesMgr:1.0       mzzsrv                      oracle-sun         1
IDL:ves/fcty:1.0          mkcf                       oracle-sun        1
IDL:ves/sts:1.0           mkcf                       oracle-sun        1
IDL:ydch/och:1.0         mnorbsrv                    oracle-sun         1
IDL:ydim/imr:1.0         mnorbsrv                    oracle-sun         1
```

Configure and test the network connection

The recommended configuration contains a public network interface which directly connects to display clients. A second private network interfaces between OVS and the ves-ready encoder. The Oracle Media Net address (OMN_ADDR) should be configured with the network IP address of the OVS public network interface.

Figure 8–2 NIC Connections





The above figure illustrates a typical setup in which the video clients and NIC-A are in one family of network addresses, and the encoder and NIC-B are in a separate family. Since OVS is connected to two networks, it is very important to make sure that data packets can be routed from one network to another. Use the **ping** utility[1] and follow these steps to test for proper physical connection:

1. On the encoder, **ping** the NIC-B network interface. If this fails, check the physical connections and IP addresses. If it then performs successfully, continue to step 2.
2. On the encoder, **ping** the NIC-A network interface. If this operation fails, the routing table is not set up correctly. Check the routing table on the server and also verify that the default gateway on the encoder is set to NIC-B.
3. Use the **mnping** utility to verify connection to Oracle Media Net. If this operation fails, make sure the server is running and the environment variable OMN_ADDR on the encoder is set. A sample response is shown below:

[1] For further information on using the ping utility, see "Network Packet Size" in Chapter 2.

```
% mnping
> echo from 0xffff ffff.1, ms=1
```

Start a real-time feed session

Refer to the user documentation for your encoder for information on configuring and initiating a Real-time Feed.

Shutting down the feed session

Should you need to stop **vsfeedsrv** without bringing down OVS, use **mnorbadm**:

```
% mnorbadm -n vsfeedsrv -s down
```

For Further Information

For further information on the Real-time Feed service, refer to the following documents and sections:

- *Introducing Oracle Video Server*
- Chapter 7, “Extending Video Encoders for Real-time Feeds”, in the *Oracle Video Server Developer’s Guide*
- Appendix G,“Setting Up OVS Sample Applications”, in the *Oracle Video Server Developer’s Guide*

9

Configuring the Logical Content and Scheduling Services

This chapter discusses the Logical Content and Scheduling Services, creating a database account for Oracle Video Server's use, and configuring the Logical Content and Scheduling Services for optimum performance.

> **Note:** This chapter assumes you are familiar with administering an Oracle database. For information on Oracle database administration, refer to the documentation supplied with your release of the Oracle database.

Topics in this chapter are:

Understanding the Logical Content and Scheduling Services

Oracle Video Server requires the use of a database to enable certain optional services:

- Logical Content Service
- Scheduling Service

The Logical Content Service

Oracle Video Server allows you to control and mange your video and audio data by storing information about these files in an Oracle database. This service, called Logical Content, allows greater flexibility and control in managing your media data, and in building media enabled applications.

For a complete description of Oracle Video Server's content model, see the *Oracle Video Server Content Administrator's Guide.*

Running without a Database

If you are running Oracle Video Server without a database, the content model changes. When Oracle Video Server is started without a database, the content service obtains content data directly from the tag file headers stored in the MDS file system. Thus there are no clips or logical content titles in a system without a database.

The Scheduling Service

The Scheduling Service provides broadcasters with a mechanism to schedule start-times and end-times for programs at regular intervals on channels you specify. For example, you might specify that Oracle Video Server play a program every fifteen minutes. This allows the viewer to tune in to the beginning of a program at fifteen minute intervals. The Scheduling Service can be used for television broadcasting, near video-on-demand, pay-per view television, and LAN multicasting.

Database Requirements

To use the Logical Content and Scheduling Services, install Oracle Video Server with an Oracle database and create a user account for Oracle Video Server. You can create the database account during the installation of Oracle Video Server, or you can create it after initial software installation with the **vsdbbuild** utility.

The database-enabled services require the following software products:

- Oracle database server 7.3.3.1 or later. In Production environments the database must be run on a separate server computer with a TCP/IP connection to the server running Oracle Video Server. You must enable XA transaction processing.
- Oracle Video Server Manager (VSM), a graphical, Java-based management tool used to create logical content titles and schedule broadcast events. Install VSM from the Oracle Video Server product distribution CD-ROM. See the *Oracle Video Server Installation Guide* for more information.

Creating A Database Account

If you created a database account during the installation of Oracle Video Server, you can use Oracle Video Server Manager to create logical content titles. For information on creating logical content titles and scheduling program events, refer to:

- *Oracle Video Server Content Administrator's Guide*
- Oracle Video Server Manager online Help

The **vsdbbuild** utility creates the user account for the Logical Content and Scheduling Services. **vsdbbuild** allows you to:

- create a new database user account for Oracle Video Server
- populate an existing database account with Oracle Video Server database objects
- remove (drop) a user account and recreate it with new Oracle Video Server tables

vsdbbuild accepts the following input to create the user account:

- Oracle system account password
- user name and password for the database account
- TNS connect alias

- name to use for the Oracle Video Server default tablespace
- name to use for the Oracle Video Server temporary tablespace

Before using **vsdbbuild** to create the user account, ensure that you have the correct information to create the database account under which you will operate Oracle Video Server. If you are not using a Global Name Server for database connectivity, ensure that the database to which you wish to connect is defined in your **tnsnames.ora** file before creating the database account with **vsdbbuild**. Obtain this information from your organization's database administrator.

Privileges Required to Create A User Account

To ensure sufficient privileges to create a user account, the **vsdbbuild** utility requires that you supply the Oracle system account password when creating or removing a user account.

Creating the User Account

The syntax to create a user account is:

```
% vsdbbuild -s system-password -u data-tablespace -t temp-tablespace username/password@alias
```

where:

-s *system-password*
specifies the Oracle system account password.

-u *data-tablespace*
specifies the name to use for the default Oracle Video Server ***data-tablespace***. When not specified, **vsdbbuild** uses the default tablespace name of USERS. Note that **-u** can be used only when creating a new user account with the **-s** option.

-t *temp-tablespace*
specifies the name to use for the temporary tablespace when creating a new user account. When not specified, **vsdbbuild** uses the default temporary tablespace name of TEMP. Note that you can only specify the **-u** option when creating a new user account with the **-s** option.

username/password@alias
The connect string identifies the user account to create or modify and the database to use for that account.

The syntax of the connect string is:

username
The user name the account is created for.

password
The password to use with the account.

alias
The TNS connect alias identifying the database.

Example 9–1 Creating a New User Account

```
% vsdbbuild -s manager -u users -t temp ovs/ovs@ovsdb
```

This example creates a user account with the following attributes:

system-password	**manager**
default tablespace name	**users**
temporary tablespace name	**temp**
user name	**ovs**
password	**ovs**
TNS connect alias of the database	**ovsdb**

After you create the database account, edit the file **$ORACLE_HOME/vs30/admin/mnrc**. The **mnrc** file serves as a repository for certain Oracle Video Server configuration information. You will find more information on editing the **mnrc** file in the section Creating a Database Connection.

Example 9–2 Creating a Schema for an Existing User Account

```
% vsdbbuild ovs/ovs@ovsdb
```

This example populates the Oracle Video Server database account for the user connect string **ovs/ovs@ovsdb**.

Creating a Database Connection

To create a database connection between Oracle Video Server and the Oracle database, edit the file **$ORACLE_HOME/vs30/admin/mnrc** to include database connection information for your system. The **mnrc** file serves as a repository for database connection information. The processes which use the **mnrc** file to connect to the database are:

- **vscontsrv**
- **vsbcastsrv**

Editing the File

The **mnrc** file is pointed to by the environment variable **$YSRESFILE**. Ensure that **$YSRESFILE** is properly set using the **echo** command:

```
% echo $YSRESFILE
$ORACLE_HOME/vs30/admin/mnrc
```

This example shows a sample **mnrc** file with entries for **vscontsrv** and **vsbcastsrv**.

```
ys.log.msg-path=/home/oracle/vs30/mesg
ys.log.msg-path=/home/oracle/mn33/mesg
vscontsrv.connect=ovs/ovs@ovsdb
vsbcastsrv.connect=ovs/ovs@ovsdb
```

The syntax for the connect string is:

username/password@alias

where:

username	user account ID for the database storing logical content information
password	password for the database account storing logical content information
alias	TNS connect alias for your database installation. For a complete discussion on SQL*Net connectivity, refer to the *Oracle SQL*Net Administration Guide*.

Enabling the Logical Content Service

To enable the Logical Content Service, simply create a database account and establish a connection with the database as described in the sections:

- Creating A Database Account
- Creating a Database Connection

By default, the content service (**vscontsrv**) is installed without a connection to a database. Once you create a database account and specify a database connect string for **vscontsrv**, it will establish a connection with the database and allow you to register your content.

For information on registering content with the database, refer to the *Oracle Video Server Content Administrator's Guide*.

Enabling the Scheduling Service

The Scheduling Service consists of the following processes:

- **vsnvodsrv**
- **vsschdsrv**
- **vsbcastsrv**

Depending on the options you choose during installation of the Oracle Video Server software, you may need to enable the Scheduling Service. To enable the Scheduling service:

- Use Oracle Video Server Manager to specify that the service starts

or

- Edit the Scheduling Service processes in the **ovsstart** file.

This section discusses editing the **ovsstart** file to start the Scheduling Service. If you wish to use VSM to enable the Scheduling Service, refer to the Oracle Video Server Manager online Help.

Note: If you choose to enable either the Logical Content or Scheduling Services using VSM, you cannot modify the parameters determining database connectivity. Depending on the user requirements placed on the system, this may adversely affect performance and scalability.

If you need to modify the parameters controlling database connectivity, modify the file **$ORACLE_HOME/vs30/admin/ovsstart** and select this as your startup file from the VSM console. For information on selecting a different VSM startup file, refer to the Oracle Video Server Manager online Help.

If you did not enable the Scheduling Services during the installation process, the **ovsstart** entries appear as shown below:

```
#
# echo -n "starting broadcast data service.."
# $ORACLE_HOME/bin/vsbcastsrv &
# sleep 5

# echo -n "starting NVOD exporter ........ "
# $ORACLE_HOME/bin/vsnvodsrv &
# sleep 2

# echo -n "starting scheduler ............ "
# $ORACLE_HOME/bin/vsschdsrv &
# sleep 2
```

To enable the processes, remove the pound sign (#) from the process entries as shown:

```
#
echo -n "starting data service.. "
$ORACLE_HOME/bin/vsbcastsrv &
sleep 5

echo -n "starting NVOD exporter........ "
$ORACLE_HOME/bin/vsnvodsrv &
sleep 2
```

```
echo -n "starting scheduler..... "
$ORACLE_HOME/bin/vsschdsrv &
sleep 2
```

Configuring Database Sessions

Depending on the user demands placed on your system, you may need to configure database-enabled services to support a greater number of users. How you configure these services depends on the user demands placed on your system.

Understanding Database Sessions

A *database session* is a specific connection of a user to the Oracle database. In order to function properly, you must configure the database services to support the number of concurrent users your Oracle Video Server system supports. The parameter which determines the number of concurrent database sessions is ***threads***

Calculating Database Sessions

The formula to calculate the number of database sessions is:

(*1 logon session + 1 commit/rollback session + 1 cursor*) × *threads* = *number of sessions*

where:

threads	The maximum number of threads **vscontsrv** can use to access the database. The default value is one.
number of sessions	The number of database sessions **vscontsrv** can open with the Oracle database. Using the default thread value of one **vscontsrv** can establish three concurrent database sessions.

Increasing Database Sessions

To increase the number of database sessions **vscontsrv** can establish with the database, increase the number of threads with the **-n *threads*** option. This example specifies five threads.

Example 9–3 Configuring threads with vscontsrv

```
% vscontsrv -n 5
```

When not specified, **vscontsrv** defaults to 1 thread. This allows **vscontsrv** to support 3 concurrent database sessions.

Configuring the Scheduling Service

The Scheduling Service uses three parameters to configure performance and scalability. The parameters are:

- how often the scheduler (**vsschdsrv**) "wakes up" to read event schedules from the database
- number of threads the exporter (**vsnvodsrv**) uses. This value is related to the number of concurrent events you schedule.
- adjusting the ***setuptime*** parameter to account for latencies in the system.

Configuring the Scheduler

The scheduler (**vsschdsrv**) "wakes-up" at intervals you specify and reads program schedule information from the Oracle database server. How often **vsschdsrv** reads program information from the database depends on the requirements of the system. Consider the following issues when setting the wake-up time of the scheduler:

- specifying too great a wake-up interval limits your scheduling options. For example, if you need to modify a program's start time to insert special programming (a currently breaking news item, for example), then a wake-up interval of 60 minutes or more may be unacceptable.
- specifying too short a wake-up interval may place an unacceptable performance load on the system. Every time **vsschdsrv** wakes-up, it reads the schedule information from **vsbcastsrv**.

By default, the **vsschdsrv** process uses a wake-up interval of 60 seconds. For most systems, this is too frequent a wake-up interval. A more appropriate value might be 10 or 15 minute intervals. Such a wake-up interval places a manageable performance load on the system, while allowing schedule changes at reasonable intervals.

Example 9–4 Configuring the NVOD scheduler (vsschdsrv)

```
% vsschdsrv -t 600
```

The **-t** option specifies how often **vsschdrv** "wakes up." This example specifies a wake-up interval of 600 seconds (10 minutes).

Configuring the Exporter Service

To function properly, the exporter service (**vsnvodsrv**) must use an appropriate number of threads to handle multiple programs scheduled to start at the same time.

The number of threads you specify for **vsnvodsrv** is determined by the number of programs scheduled to begin or end at the same time. For example, if you have ten channels, you should configure **vsnvodsrv** to use 10 threads.

Example 9–5 Specifying the number of threads for scheduled events

```
% vsnvodsrv -n 10
```

This example specifies 10 threads with the **-n** option. When not specified, **vsnvodsrv** defaults to three threads.

Configuring Setup Time

To ensure that your programs begin playing on time, you must take into account the amount of *setup time* required for all necessary play preparation to take place. Setup time is the time required to:

- read program information from the database
- establish a downstream connection with the client
- Oracle Video Server's internal response time to both read the information from the database and allocate downstream connection (generally 1 to 2 seconds)

Consider these issues carefully when calculating the setup time.

Setup Time Example

This example is designed to help you understand the issues involved in determining system latencies. Note that values for your Oracle Video Server system will be different from those presented in this example. You must determine values for your specific system in order to calculate an accurate setup time value.

Suppose you wish to begin playing a program at 8:00 pm. System performance tests indicate the following system and network latencies:

- The database requires two seconds to read program title and schedule information.
- The network in use requires three seconds to establish a downstream connection with the client device.

- Oracle Video Server requires two seconds to read the schedule information from the database and allocate a downstream connection.

In this example the system requires seven seconds of setup time to begin playback at exactly 8:00 pm.

The setup time is controlled by the ***setuptime*** parameter stored in the EXPORTER table of the database. You can modify this value using the following SQL statement:

Example 9–6 Modifying the setuptime parameter

```
SQL> update exporter set setuptime = 7000000 where implid = 'vsnvodsrv';
```

The ***setuptime*** parameter is specified in microseconds (1 second=1 million microseconds). By default, **vsnvodsrv** uses a setup time of three seconds (3 million microseconds).

10

Configuring the Session-and-Circuit Service

The Session-and-Circuit Service establishes and maintains client/server communication channels and manages a set of Oracle Video Server resources on behalf of a particular client device. You can create a configuration file to map server resources to specific network addresses, improving Oracle Video Server's ability to balance its network load. This chapter describes the syntax of the session-and-circuit configuration file, and provides example configurations.

Topics in this chapter are:

Understanding the Session-and-Circuit Service

When a client sends a request to Oracle Video Server, the session-and-circuit manager (**vscsmsrv**) uses information supplied by the client to assign an appropriate video pump (also called a *channel provider*) to that client. By default, any client request, on any network segment, is assigned the first available video pump that fits the client's requested parameters.

The session-and-circuit configuration file allows video pumps (or other channel providers) to be mapped to specific client groups (clients belonging to a particular network segment), you can constrain client requests to video pumps to better deal with segmented network topologies. The primary advantage in creating such a configuration file is that it allows Oracle Video Server to use multiple network interface cards (NICs), providing increased network bandwidth and load balancing.

The session-and-circuit configuration file allows Oracle Video Server to make "more intelligent decisions" when allocating a video pump to a client by providing the Session-and-Circuit Service with more information about a video pump and its capabilities.

The configuration file also provides the ability to deliver multiple network protocols from a single server. For example, a channel provider could use two network interfaces:

- ATM interface providing both ATM AAL5 and IP over ATM
- Ethernet interface providing IP

Different protocols can be assigned to different network segments as appropriate.

Using a Configuration File to Define the Session-and-Circuit Service

To use a configuration file with the Session-and-Circuit Service:

1. Create a configuration file mapping (or constraining) clients on specific network segments to specific video pumps.
2. Choose item **a** or **b** depending on the administrative interface you are using:
   a. Use the VSM console to specify a file location from the Startup Preferences menu.
   b. Edit the **vscsmsrv** entry of the file **$ORACLE_HOME/vs30/admin/ovsstart** and specify that it load the configuration file at Oracle Video Server start up. For example:

      ```
      vscsmsrv -f file:$ORACLE_HOME/vs30/admin/vcm.cfg
      ```

Creating a Session-and-Circuit Configuration File

You must obtain the following information to create the session-and-circuit configuration file:

- server hostname.
- device names of network interface cards available to Oracle Video Server.
- maximum continuous bandwidth that can reliably be provided by the individual network interfaces. For example, suppose you have a Fast-Ethernet network card with a maximum bandwidth rating of 100Mbps, which represents the maximum bandwidth that can be obtained during brief moments or *bursts*. The *sustained* bandwidth of a network card is generally 60–70% of the maximum *burst* rate. In the case of the Fast-Ethernet interface this is approximately 60 Mbps.
- network protocols and their associated segments. Typical protocols include UDP, TCP, and ATM. Note that ATM (AAL5-SVC or AAL5-PVC) is not supported on all platforms. For information on supported protocols, refer to the *Release Notes* supplied with your software release.
- client addresses. Specified as subnet addresses or use a regular expression to specify only certain network addresses.

- names of video pumps. The -o option of the video pump assigns a label (name) to a specific instance of the video pump. For information on assigning a label to a video pump, refer to the vspump in Chapter 11, “Oracle Video Server Components”.

Parts of the Configuration File

The session-and-circuit configuration file consists of three main components:

- network interfaces
- communication links
- client groups

Network Interfaces

The *network interface* portion of the configuration file includes information about the server and the network interfaces available to Oracle Video Server. The information used to define the network interface is:

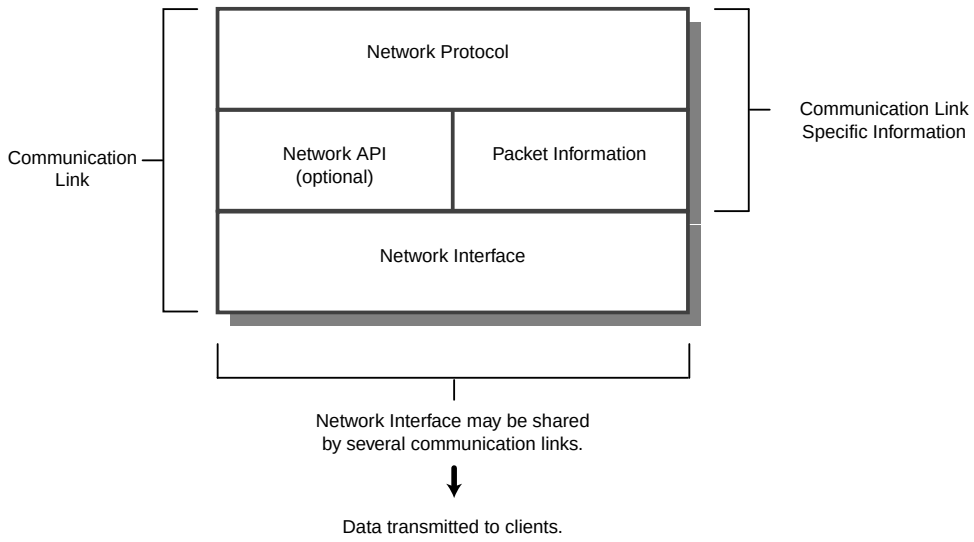
- network device names
- server hostname
- sustained bandwidth available from the individual network cards

Communication Links

The *communication links* portion of the configuration file defines paths of communication between the video pump processes and the network interfaces available to Oracle Video Server. The video pump uses communication links to deliver data to a client. The information used to define a communication link consists of:

- network protocol. The network protocol the system uses to send data. Typically this is UDP, TCP, or ATM (AAL5-SVC or AAL5-PVC).
- API used to communicate with the network interface card. This information is optional. To obtain information about the software API used by your network card, refer to the documentation provided by your network card vendor.
- packet size. This is the packet size specified with the **-n** option of the video pump. Typical packet sizes are 8192 bytes (8 K) or 1024 (1 K) bytes. For information on the packet size used by the video pump, refer to the **vspump** section in *Part II* of this guide.
- name of the network interface.

Figure 10–1 Parts of the communication link



Client Groups

The *client group* portion of the configuration file defines collections of clients and the video pumps that are compatible with them. Any video pump can service any client within its client group. The information used to define a client group is:

- label identifying the client group
- string (or series of strings) identifying the video pump. You identify a video pump by assigning a label with **-o** line option. Refer to the **vspump** section in *Part II* of this guide for more information on assigning a label to a video pump.
- communication link(s) to use for a given client group
- clients that belong to the client group. Clients are identified by the network protocol in use for the client group and a regular expression providing addresses of the clients.

Examples of Session-and-Circuit Configuration Files

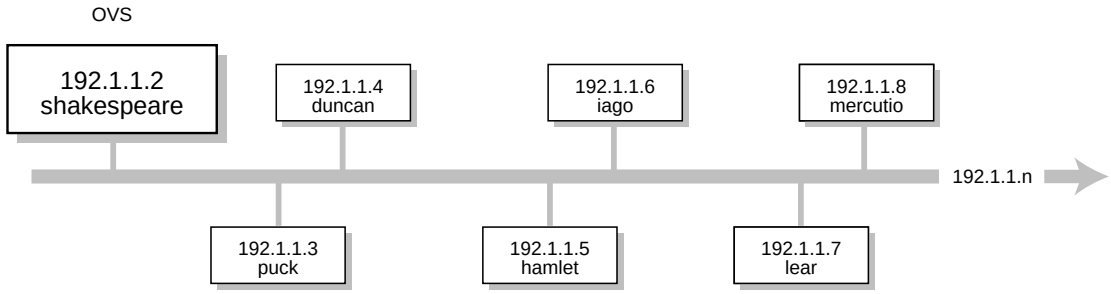
The following examples are designed to provide you with enough information to create a session-and-circuit configuration file for your own Oracle Video Server system. The examples consist of a brief scenario followed by a network diagram and a configuration file appropriate to the example network. The examples are:

- Example 1: Mapping a Single Subnet Address
- Example 2: Using Two Network Interfaces and Two Video Pumps
- Example 3: Using a Regular Expression to Specify Network Addresses

Example 1: Mapping a Single Subnet Address

This example creates a configuration file for a server using a single network interface to provide data to a single subnet address. The network is diagrammed in Figure 10–2.

Figure 10–2 Network diagram for Example 1



Network Interface for Example 1

The network interface for Example 1 is a Fast-Ethernet card with the device name **/dev/en0**. Although Fast-Ethernet cards are rated at 100Mbps maximum throughput, a more realistic rate for streaming video is 60Mbps (60% of the total available bandwidth.) This ensures that the network card has overhead available to allow continuous throughput. The hostname of the server computer is **shakespeare** and the label assigned to the interface is **interface0**.

```
NIC interface0 { shakespeare /dev/en0 60000000 };
```

The syntax for ***network interface*** is:

<network-info> = 'NIC' *<interface-name>* '{' <server-*hostname*> *<device-name>* *<maxbw>* [*<interface-info>*] '}' ';'

Communication Link Information for Example 1

The *communication links* parameter defines the communication links available to channel providers such as the video pump. Channel providers are services which establish communications links between Oracle Video Server and clients.

The communication link for this example uses **interface0** (defined above) as its network interface. This link uses UDP as the network protocol to serve clients on subnet **192.1.1**. A packet size of **8192** bytes (8k) is the specified network packet size. The packet size must be the same as that specified with the **-n** option of the video pump. For more information on specifying a packet size with the video pump refer to the vspump section of Chapter 11, “Oracle Video Server Components”.

Optionally, you can include the software API in use by the network card; this example uses BSDSOCKETS. If you are unsure of the software API in use by your server’s network card you may leave this parameter out. The name assigned to this communication link is **linkA**.

```
LINK linkA { interface0 UDP 192.1.1.* 8192 8192 8192 BSDSOCKETS };
```

The syntax for ***communication links*** is:

link-information = 'LINK' *<linkname>* '{' *<interface-name>* *<protocol>* *<address>* *<pktpref>* *<pktmax>* *<pktmod>* [*<api-name>*] [*<api-info>*] '}' ';'

Client Group Information for Example 1

The *client groups* parameter maps (or constrains) clients on specific network segments to use the specified video pumps. This ensures that a given video pump has appropriate resources to serve clients belonging to a particular client group. If a client sends a request to Oracle Video Server, but is not included in the configuration file, **vscsmsrv** assigns it the first available video pump which meets the parameters of the client request. Such a client is a member of the *default group*. The default group allows clients with network addresses not included in the session-and-circuit configuration file to connect to Oracle Video Server.

The *client group* parameter is divided into two segments:

- PROVIDERS—channel providers such as the video pump and the network protocol in use by that channel provider.
- CLIENTS—network addresses that the video pump will establish a session with.

In this example, the video pump (labeled **vspump0)** provides data using the UDP protocol to clients residing on subnet **192.1.1**. The client group is labeled GROUPA.

```
GROUP GROUPA
{
     PROVIDERS
    {
     vspump0 linkA;
    }
     CLIENTS
    {
     UDP 192.1.1.*;
    }
};
```

The syntax for the *client group* is:

```
<client-group> = 'GROUP' <groupname> '{' 'PROVIDERS' '{' (<pvdrname> <linkname>
';')+ '}' 'CLIENTS' '{' (<protocol-name> <address> ';')+ '}' '}' ';'
```

Configuration File for Example 1

This is the complete configuration file used in Example 1. To better understand the structure of the file, review the comments included for each parameter.

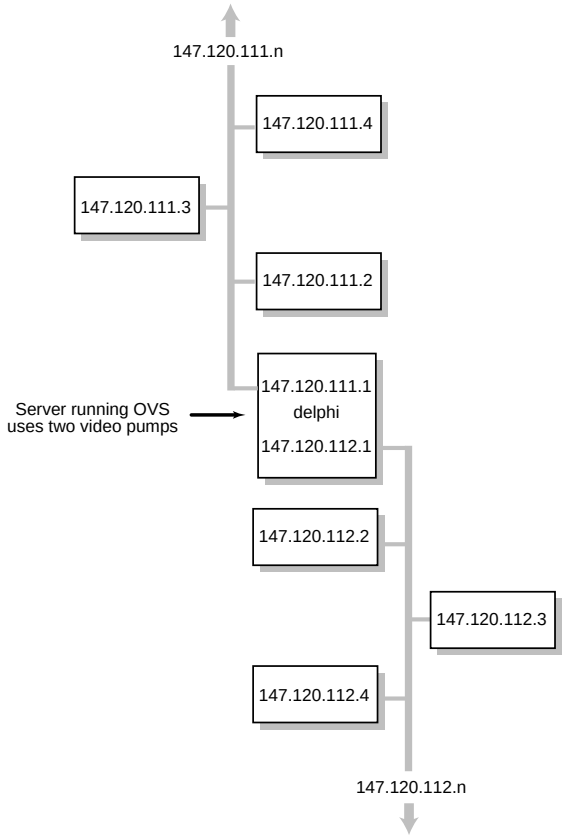
Example 10–1 Configuration file for Example 1

```
//Network Interface Card (NIC)
//interface-name=interface0
//server hostname=shakespeare
//device name=/dev/en0
//maxbw available from en0=60000000 (60 Mbps)
NIC interface0 { shakespeare /dev/en0 60000000 };
//
//linkname=linkA
//interface-name=interface0
//protocol=UDP
//network address=192.1.1.*
//preferred packet size=8192 (8k)
//maximum packet size=8192 (8k)
//minimum packet size=8192 (8k)
//API name=BSDSOCKETS
LINK linkA { interface0 UDP 192.1.1.* 8192 8192 8192 BSDSOCKETS };
//
//channel group-name=GROUPA
//provider-name=vspump0
//link-name=linkA
//Clients for vspump0 are located on subnet 192.1.1.*
//protocol=UDP/IP
//subnet address=192.1.1.*
//
GROUP GROUPA
{
     PROVIDERS
    {
     vspump0 linkA;
    }
     CLIENTS
    {
     UDP 192.1.1.*;
    }
};
```

Example 2: Using Two Network Interfaces and Two Video Pumps

This example creates a configuration file for a server using two network interfaces and two video pumps. Each video pump is assigned to an interface which in turn maps to clients on one of two network subnets (**147.120.111** and **147.120.112**). The network is depicted in Figure 10–3.

Figure 10–3 Network diagram for Example 2



Network Interface Information for Example 2

The server **delphi** uses two FDDI (Fiber Distributed Data Interface) network cards; one for each subnet to which video is provided. Note that the sustained bandwidth is listed as 70 Mbps. Although FDDI is rated at 100 Mbps a more realistic continuous throughput is 70 Mbps.

```
NIC nic0 { delphi /dev/fddi0 70000000};
NIC nic1 { delphi /dev/fddi1 70000000 };
```

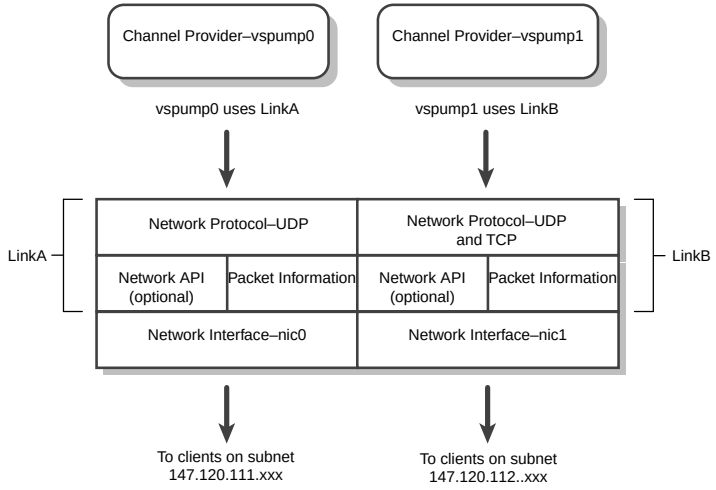
Communication Link Information for Example 2

Oracle Video Server uses two communication links:

- **linkA** serves subnet 147.120.111 and provides UDP/IP
- **linkB** serves subnet 147.120.112 and provides UDP/IP and TCP/IP

```
LINK linkA { nic0 UDP 147.120.111.* 8192 8192 8192 };
LINK linkB { nic1 UDP 147.120.112.* 8192 8192 8192 };
LINK linkB { nic1 TCP 147.120.112.* 8192 8192 8192 };
```

Figure 10–4 Communications links for Example 2



Client Group Information for Example 2

Two client groups are defined:

- **group0** consists of **vspump0** delivering UDP to subnet 147.120.111.
- **group1** consists of **vspump1** delivering UDP and TCP to subnet 147.120.112.

```
GROUP group0
{
     PROVIDERS
     {
      vspump0 linkA;
     }
     CLIENTS
     {
      UDP 147.120.111.*;
     }
};
GROUP group1
{
     PROVIDERS
     {
      vspump1 linkB;
      vspump1 linkB;
     }
     CLIENTS
     {
      UDP 147.120.112.*;
      TCP 147.120.112.*;
     }
};
```

Configuration File for Example 2

This is the complete configuration file used in Example 2. To better understand the structure of the file, review the comments included for each parameter.

Example 10–2 Configuration file for Example 2

```
//Server Network Interface Card (NIC)
//NIC name=nic0
//server hostname=delphi
//NIC device name=/dev/fddi0
//NIC bandwidth (sustained)=70 Mbps
NIC nic0 { delphi /dev/fddi1 70000000};
//
//NIC name=nic1
//server hostname=delphi
//NIC device name=/dev/fddi1
//NIC bandwidth (sustained)=70 Mbps
//
NIC nic1 { delphi /dev/fddi1 70000000};
//
//Communications link--defines communications links available to channel
// providers such as vspump.
//linkname=linkA
//network protocol and address=147.120.111.*, UDP/IP
//NIC name=nic0
//preferred packet size=8192 (8k)
//maximum packet size=8192 (8k)
//minimum packet size=8192 (8k)
//API name=BSDSOCKETS
//
//LINK linkA { nic0 UDP 147.120.111.* 8192 8192 8192 };
//
//linkname=linkB
//network protocol and address=147.120.112.*, UDP/IP and TCP/IP
//NIC name=nic1
//preferred packet size=8192 (8k)
//maximum packet size=8192 (8k)
//minimum packet size=8192 (8k)
//API name=BSDSOCKETS
//
LINK linkB { nic1 UDP 147.120.112.* 8192 8192 8192 };
LINK linkB { nic1 TCP 147.120.112.* 8192 8192 8192 };
//
```

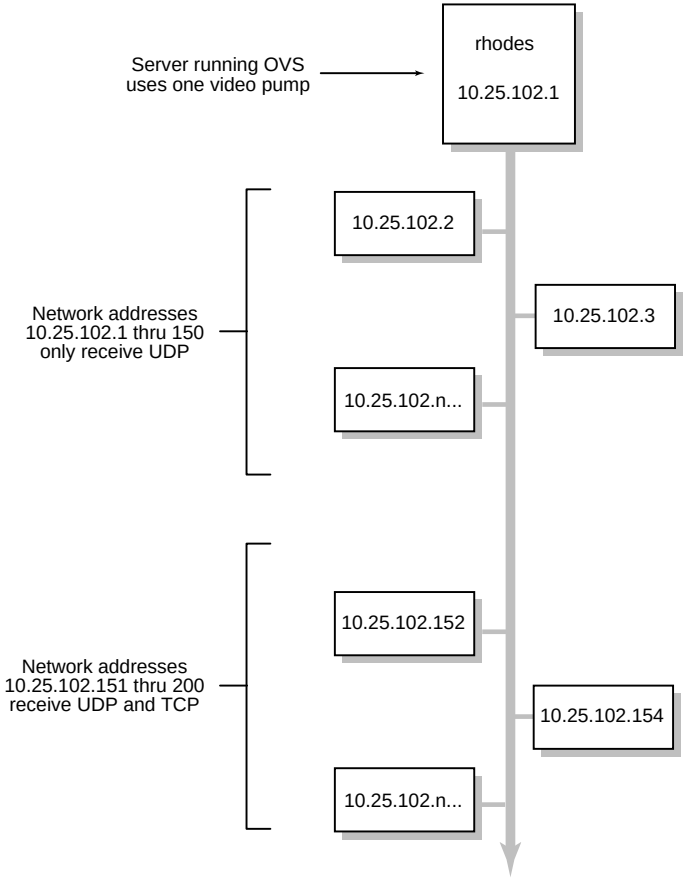
```
//Channel Group
//Defines channel group for IP subnet 147.120.111.*
//channel provider for GROUP group0 consists of vspump0 using UDP
//Clients for vspump0 are located on IP subnet UDP 147.120.111.*
//vspump0 only provides UDP
//Clients for vspump0 are located on IP subnet UDP 147.120.111.*
//vspump0 only provides UDP
//
GROUP group0
{
     PROVIDERS
     {
      vspump0 linkA;
     }
     CLIENTS
     {
      UDP 147.120.111.*;
     }
};
//
//Defines channel group for IP subnet 147.120.112.*
// channel provider for GROUP group1 consists of vspump1 using UDP and TCP
//Clients for vspump1 are located on IP subnet UDP 147.120.112.*
//vspump1 will deliver either UDP or TCP encapsulated media streams
//
GROUP group1
{
     PROVIDERS
     {
      vspump1 linkB;
      vspump1 linkB;
     }
     CLIENTS
     {
      UDP 147.120.112.*;
      TCP 147.120.112.*;
     }
};
```

Example 3: Using a Regular Expression to Specify Network Addresses

The Session-and-Circuits Service allows you to specify network addresses of specific clients using regular expressions. This allows you to constrain a group of clients to a specific client group. In this example:

- subnet 10.25.102 receives UDP
- network addresses 10.25.102.151 through 10.25.102.200 receive UDP and TCP.

Figure 10–5 Network diagram for Example 3



Network Interface for Example 3

This example uses a single Fast-Etherent interface (**interface0**) to provide 60Mbps of network bandwidth. The server hostname is **rhodes** and the device name of the Fast-Ethernet card is **/dev/en0**.

```
NIC interface0 { rhodes /dev/en0 60000000 }
```

Communication Link Information For Example 3

Example 3 uses two communication links:

- **linkUDP** provides UDP to subnet 10.25.102
- **linkTCP** provides TCP only to clients on network addresses 10.25.102.151 through 10.25.102.200. See the example for the regular expression used to specify these network addresses.

Note that both communication links share the same network interface (**interface0**).

```
LINK linkUDP { interface0 UDP 10.25.102.* 8192 8192 8192 };
LINK linkTCP { interface0 TCP 10.25.{102.15[1-9],1[6-9][0-9],200} 8192 8192 8192 };
```

Client Group Information For Example 3

The client group for this configuration consists of a single group (**GROUPA**) using the two previously defined communication links (**linkUDP** and **linkTCP** respectively). Since the network addresses using TCP are defined in the communication-link syntax, you can specify the subnet 10.25.102 for both the UDP and TCP links and be assured that only those addresses specified in the syntax for **linkTCP** will receive TCP.

```
GROUP GROUPA
{
     PROVIDERS
    {
      vspump0 linkUDP;
      vspump0 linkTCP;
    }
     CLIENTS
    {
      UDP 10.25.102.*;
      TCP 10.25.102.*;
    }
};
```

Configuration File for Example 3

This is the complete configuration file used in Example3. To better understand the structure of the file review the comments included for each parameter.

Example 10–3 Configuration file for Example 3

```
//Network Interface Card (NIC)
//interface-name=interface0
//server hostname=rhodes
//device name=/dev/en0
//maxbw available from en0=60000000 (60 Mbps)
//
NIC interface0 { rhodes /dev/en0 60000000 };
//
//linkname=linkUDP
//interface-name=interface0
//protocol=UDP
//preferred packet size=8192 (8k)
//maximum packet size=8192 (8k)
//minimum packet size=8192 (8k)
//API name=BSDSOCKETS
//
LINK linkUDP { interface0 UDP 10.25.102.* 8192 8192 8192 };
//
//linkname=linkTCP
//interface-name=interface0
//protocol=TCP
//preferred packet size=8192 (8k)
//maximum packet size=16384 (16k)
//minimum packet size=1024 (1k)
//API name=BSDSOCKETS
//
LINK linkTCP { interface0 TCP 10.25.{102.15[1-9],1[6-9][0-9],200} 8192 8192 8192 };
//
```

```
//GROUP=GROUPA
//channel provider=vspump0
//link-name=linkUDP
//nwtwork prtocol=UDP and TCP
//Clients for vspump0 are located on subnet 10.25.102.*
//clients can receive UDP and TCP
//subnet address=10.25.102.*
//
GROUP GROUPA
{
     PROVIDERS
    {
      vspump0 linkUDP;
      vspump0 linkTCP;
    }
     CLIENTS
    {
      UDP 10.25.102.*;
      TCP 10.25.102.*;
    }
};
```

The Default Configuration

The **vscsmsrv** service includes a built-in default configuration allowing:

- operation of the Session-and-Circuit Service without specifying a configuration file
- allows the Session-and-Circuit Service to allocate sessions to clients not included in the configuration file
- allows the Session-and-Circuit Service to allocate video pumps not included in the configuration file to requesting clients

Viewing the Default Configuration

To view the default configuration for your Oracle Video Server system:

1. Specify that **vscsmsrv** start without a configuration file. Depending on the administrative interface you use to configure your system:

   a. Use the VSM console to delete any configuration file entry

   or

   b. Edit the **vscsmsrv** entry of the file **$ORACLE_HOME/vs30/admin/ovsstart** and specify that vscsmsrv start without a configuration file. For example:

   ```
   % vscsmsrv &
   ```

2. Start Oracle Video Server.
3. View the default configuration using **vscsmdir** (session and circuit list utility) with the **-g** option:

   ```
   % vscsmdir -g
   ```

An Example Session-and-Circuit Default Configuration

In this example, the default configuration is based on an Oracle Video Server system using two video pumps (**vspump1** and **vspump2**). The Session-and-Circuit Service creates two communication links for each video pump. One communication link provides UDP, the other TCP.

Interface Information for the Default Configuration

The default interface is labeled **default-nic**. The server hostname and network device are not specified. The **vscsmsrv** service assumes a bandwidth adequate for any reasonable number of requesting clients using such a configuration (4 GB).

```
NIC default-nic { <unspecified> <unspecified> 4294967295 };
```

Communication Link Information for the Default Configuration

Each available video pump is assigned two communication links: one providing UDP and one providing TCP. The communication links are numbered sequentially in the order they are started (i.e. 0001, 0002, 0003, 000*n* ...). This example uses two video pumps and shows the four communication links such a configuration yields. All default-links use the default interface (**default-nic**). The packet size used is that

specified with the **-n** option of **vspump**. The **vscsmsrv** service reads this information from the video pumps at server startup.

```
LINK default-link-0000 { default-nic UDP * 8192 8192 8192  };
LINK default-link-0001 { default-nic TCP * 8192 8192 8192  };
LINK default-link-0002 { default-nic UDP * 8192 8192 8192  };
LINK default-link-0003 { default-nic TCP * 8192 8192 8192  };
```

Client Group Information for the Default Configuration

The default configuration creates a single client group called DEFAULT_GLOBAL. The DEFAULT_GLOBAL group assigns all available video pumps to any requesting client. When a client makes a request, **vscsmsrv** matches the request to the first available video pump which meets the requesting clients parameters. If, for example, a client sends a request that it requires a TCP connection, and **vspump1** is available, a session will be allocated using communication link **default-link-0001**. This is because **default-link-0001** is the first link with an available video pump using **TCP.**

```
GROUP DEFAULT_GLOBAL
{
     PROVIDERS
     {
      vspump2 default-link-0002;
      vspump2 default-link-0003;
      vspump1 default-link-0000;
      vspump1 default-link-0001;
     }
     CLIENTS
     {
      * *;
     }
};
```

Complete Output of the Default Configuration

Here is the default configuration used in Example 4:

Example 10–4 Default configuration for Example 4

```
% vscsmdir -g
NIC default-nic { <unspecified> <unspecified> 4294967295 };

LINK default-link-0000 { default-nic UDP * 8192 8192 8192 };
LINK default-link-0001 { default-nic TCP * 8192 8192 8192 };
LINK default-link-0002 { default-nic UDP * 8192 8192 8192 };
LINK default-link-0003 { default-nic TCP * 8192 8192 8192 };

GROUP DEFAULT_GLOBAL
     {
     PROVIDERS
     {
      vspump2 default-link-0002;
      vspump2 default-link-0003;
      vspump1 default-link-0000;
      vspump1 default-link-0001;
     }
     CLIENTS
     {
      * *;
     }
};
```

For more information on viewing your server's Session-and-Circuit Service configuration, see Viewing the Session-and-Circuit Service Configuration.

Viewing the Session-and-Circuit Service Configuration

You can view the session-and-circuit configuration using the **vscsmdir** (session and circuit list) utility. This section describes the following viewing options:

- Viewing the Specified Configuration File
- Viewing the Running Configuration
- Viewing Channels Allocated to Video Pumps

Viewing the Specified Configuration File

To view the configuration file specified when **vscsmsrv** was started, use the **-f** command line option. If no configuration file was specified, no output is produced.

```
% vscsmdir -f
NIC interface0 { oracle-sun /dev/fddi1 600000 };
LINK linkA { interface0 UDP 144.0.0.* 8192 8192 8192 };
LINK linkB { interface0 TCP 144.0.0.* 8192 8192 8192 };
GROUP group0
{
     PROVIDERS
    {
      vspump1 linkA;
      vspump2 linkB;
    }
     CLIENTS
    }
      UDP 144.0.0.*;
      TCP 144.0.0.*;
    }
};
```

Viewing the Running Configuration

The running configuration includes the DEFAULT_GLOBAL group. The DEFAULT_GLOBAL group is the Session-and-Circuit Service's built configuration. To learn more about **vscsmsrv**'s default configuration, see The Default Configuration earlier in this chapter.

To view the running configuration file use the **-g** command line option. This example uses boldface type to highlight the portion of the running configuration used for the DEFAULT_GLOBAL:

```
% vscsmdir -g
NIC default-nic { <unspecified> <unspecified> 4294967295 };
NIC interface0 { oracle-sun /dev/fddi1 600000 };
LINK linkA { interface0 UDP 144.0.0.* 8192 8192 8192  };
LINK default-link-0000 { default-nic UDP * 8192 8192 8192  };
LINK default-link-0001 { default-nic TCP * 8192 8192 8192  };
GROUP group0
{
     PROVIDERS
     {
      vspump1 linkA;
     }
     CLIENTS
     {
      UDP 144.0.0.*;
      TCP 144.0.0.*;
     }
};
GROUP DEFAULT_GLOBAL
{
      PROVIDERS
      {
       vspump2 default-link-0000;
       vspump2 default-link-0001;
      }
      CLIENTS
      {
       * *;
      }
};
```

Viewing Channels Allocated to Video Pumps

To view video pumps registered with the Session-and-Circuit Service, and the channels allocated to them, use **vscsmdir** with the **-p** option. This provides a listing of video pumps and the number of channels allocated to each. This example lists two video pumps (**vspump1** and **vspump2**). The portion of the output highlighted in boldface text shows that:

- no channels are allocated to **vspump1** (**ch**:**0**)
- a single channel is allocated to **vspump2** (**ch**:**1**)
- the last line of the output shows that the communication link used for the connection is **link-02**, and that the link is capable of delivering a maximum bit rate of 3100000 bps (3.10 Mbps):

```
% vscsmdir -p
PRV: vspump1 ch:0(3) br:0(9300000)
   LINK link-00 { nic0 UDP * 8192 8192 8192 };
   LINK link-01 { nic0 TCP * 8192 8192 8192 };
PRV: vspump2 ch:1(3) br:0(9300000)
   LINK link-02 { nic0 UDP * 8192 8192 8192 };
   LINK link-03 { nic0 TCP * 8192 8192 8192 };
   CH link:link-02 br:3100000
```

Part II

Oracle Video Server Command Reference

t

11

Oracle Video Server Components

The following server components and configuration files make up the Oracle Video Server. These server components are started either by the VSM console or the **ovsstart** script as described in Chapter 3, “Basic Oracle Video Server Configuration Tasks”.

voltab	Configuration file that defines the Oracle Media Data Store (MDS) volume structure.
mdsdirsrv	MDS directory server. Manages the layout of files in the MDS and grants access permission to requesting clients (such as **vspump**), to create, access, or modify files. **mdsdirsrv** determines if there is bandwidth available to allow access without losing the service quality to clients already accessing the MDS.
mdsftpsrv	MDS FTP server allows an FTP client to transfer binary data between a remote host filesystem and the MDS.
mdshsmsrv	Manages the metadata pertaining to tertiary storage. This server is required to enable HSM (Hierarchical Storage Management).
mdsrmtsrv	MDS remote file server enables remote access to the MDS, including an efficient interface for downloading BLOBs.
mdsxfrsrv	Transfers data between disks and tertiary storage. This server is required to enable HSM (Hierarchical Storage Management).

vsbcastsrv	Manages broadcast data services for the Scheduling Service. **vsbcastsrv** manages schedule information from **vsschdsrv** and manages data stored in the database.
vscontsrv	Content service allows a client to query available content in the OVS system and resolves content requests coming in via the stream service.
vscsmsrv	Session-and-Circuit Manager allocates sessions and circuits for client devices connecting to OVS. See also the session-and-circuit configuration file.
session-and-circuit configuration file	Allows you to assign video pumps (or other channel providers) to specific *client groups*. A client group defines a group of clients belonging to a particular network segment. You can constrain client requests to video pumps to better deal with segmented network topologies.
vsfeedsrv	Real-time feeds server allows content to be loaded into the MDS and tagged for playback directly from an encoding station in real-time.
vsnvodsrv	Implements the scheduled playout of OVS content on specified channels. At the instruction of **vsschdsrv**, **vsnvodsrv** initiates and terminates the playout of logical content.
vsschdsrv	Generic scheduling service that reads broadcast scheduling information and keeps track of time. Working with **vsbcastsrv** and **vsnvodsrv**, it initiates broadcast playout at the correct time.
vspump	Video pump reads content files from the MDS and streams the requested data to clients over the network.
vsstrmsrv	Stream service handles requests for content files from a client, including operations such as play, pause, stop, and rewind.

Using Resource Descriptors

Every Oracle Video Server command is associated with a list of **resources**. Every resource can be set as a UNIX environment variable to define a specific behavior when a command is executed. The resource descriptor is an element in the resource database of a process. Each resource descriptor is identified by a name and may have one or more values associated with it. These values are represented as character strings.

The name of a resource uses the following syntax:

command.resource

or, if the resource has a value:

command.resource=value

Listing Resources for Commands

You can obtain a listing of all resources associated with a given command using the **-h** option to display its on-line help. For example:

```
% vscontsrv -h
usage: vscontsrv [ -hVT ] [ -c connect ] [ -m maxcursors ]
[ -n numthreads ] [ -R resource-statement ] [ -P resource-file ]

   -h -> show-usage
   -V -> print-version
   -T -> verbose=true
```

Specifying Resources

Each command has two options associated with its resources: **-R** and **-P**. The **-R** option specifies a specific resource on the command line. The **-P** option specifies a file containing resource information.

Specifying Resources on the Command Line

To set resources on the command line, use the syntax:

command -R *command.resource*

For example, to set the **verbose** resource for the **vstag** command to **true**:

```
% vstag -R vstag.verbose=true
```

Specifying Resources in a File

Multiple resources can be grouped together into a resource file. The file **$ORACLE_HOME/vs30/admin/mnrc** which is pointed to with the $YSRESFILE environment variable serves as a repository for resource information of this type. For example, the Logical Content Service (**vscontsrv**) stores database connection information in the **mnrc** file. This example shows a sample **mnrc** file with entries for **vscontsrv**:

```
ys.log.msg-path=/home/oracle/vs30/mesg
ys.log.msg-path=/home/oracle/mn33/mesg
vscontsrv.connect=oracle/ovs@ovs
```

You can use the **mnrc** file as a repository for other command resources.

voltab

The **voltab** configuration file defines the Oracle Media Data Store (MDS) volumes that **mdsdirsrv** can access. The Oracle Installer creates a **voltab** file for one volume at installation. If you require additional MDS volumes, or wish to alter your MDS configuration, you must modify this file. Before starting mdsdirsrv (MDS directory server), you must have a **voltab** configuration file.

Syntax

volume [*maxbw*={*integer*},] [*maxrate=integer*,] [*minrate=integer*,] [*dfltrate=integer*,] striped, [*width=integer*,][*noparity*,] [*raidsize=integer*,][*tocsz=integer*,][*spares=disk_name*] *disks* ...

where:

volume	the name of the MDS volume. Volume names can be from 1 to 32 characters long.
maxbw	specifies the maximum rate in megabits per second (Mbps) at which all OVS components can together read and/or write to the MDS volume.
dfltrate	specifies the default bit rate allocated to a non-real-time MDS client in Mbps. This is used whenever non-real-time MDS clients request the default bit rate.
maxrate	specifies the maximum bit rate allocated to a non-real-time MDS client in Mbps. This is used whenever non-real-time MDS clients request the maximum allowable bit rate. If **maxrate** is not specified, **dfltrate** and **minrate** are not applicable.
minrate	specifies the minimum bit rate allocated to a non-real-time client (unless the client requests less). As more clients access the MDS, the server allocates less and less bandwidth to non-real-time clients until all available bandwidth is used. The **minrate** parameter specifies the minimum bit rate that must be available for the allocation of resources to an MDS client.
striped	stripes all files in the MDS volume across all the volume's disks. MDS volumes must be striped.
width	specifies the stripe width. The default stripe width is 32K.

noparity	disables RAID protection but organizes the volume into RAID sets of the size specified by **raidsize** and performs I/O to all disks as if in a RAID set. To disable RAID protection, use a **raidsize** of 1 rather than **noparity**.
raidsize	specifies the number of disks in a RAID set. The default, and minimum, **raidsize** is 1, which disables RAID protection. A **raidsize** greater than 1 enables RAID protection. Do not use a **raidsize** greater than 6, since this would adversely affect the **maxbw** value.
tocsz	specifies the size of the volume's table of contents in raidstripes. Increasing this value increases the maximum number of files that can be stored in a volume. The size of a raidstripe is **width** x **raidsize**. The minimum, default, and recommended value is 1; there is no maximum.
spares	specifies which disk, if any, is to be used as a spare in the event of a disk failure. Each volume can have one and only one spare disk. Spares are not necessary on systems that support hot-swappable disks.
disks	specifies the disks in the volume, whose number must be a multiple of **raidsize**. Disks should be listed in one of the following ways: list of device names contained by curly braces and delimited by commas: {/dev/rdsk/c0t0d0s6,/dev/rdsk/c1t0d0s6,/dev/rdsk/c2t0d0s6} lists of numbers delimited by commas: /dev/rdsk/c{0,1,2,3}t{2,3,4,5,6}d0s{6} ranges of numbers using hyphens to span from lowest to highest digits (shorthand notation): /dev/rdsk/c{0-3}t{0-4}d0s{6}

Note: Refer to "Planning the Oracle Media Data Store" on page 2-7 for information on calculating the following **voltab** file parameters:

- spares
- disks
- raidsize
- stripe width

Usage Note

Take care when editing the **voltab** file not to introduce leading spaces before the volume name, or within the volume specification statement. Spaces can cause **mdsvolinit** to fail to recognize the volume.

Defining Multiple MDS Volumes

The *Oracle Video Server Installation Guide* for your server platform explains the proper disk notation for listing disks to be used in a volume. This example shows a **voltab** file with *multiple* volumes listed.

Example 1 This example defines two real-time MDS volumes (**volume1** and **volume2**).

```
volume1 maxbw=184,striped,width=32k,raidsize=4 /dev/rdsk/c{0-3}t{0-3}d0s{6}
volume2 maxbw=184,striped,width=32k,raidsize=4 /dev/rdsk/c{4-7}t{0-3}d0s{6}
```

Initializing the Oracle Media Data Store

Use the mdsvolinit utility to write the definition for each MDS volume in the **voltab** file to its disks. Perform this step only:

- before you start mdsdirsrv for the first time after installing the OVS software
- if you subsequently modify an MDS volume definition in the **voltab** file and need to re-initialize the volume before restarting mdsdirsrv
- if you want to erase all files from a volume

Creating an MDS Volume

To initialize an MDS volume, you must first start Oracle Media Net (OMN). OMN enables communication among the OVS components so that the **mdsvolinit** utility can write information to the MDS volume.

> **Note:** Remember to back up your **voltab** file.

To initialize an MDS volume, perform these steps:

4. Log in as the **oracle** software owner and change to the directory **$ORACLE_HOME/mn33/admin.**

   ```
   % cd $ORACLE_HOME/mn33/admin
   ```

5. Execute the OMN startup script, mnstart:

   ```
   % ./mnstart
   ```

6. Initialize the MDS volume using the **mdsvolinit** utility with the **-s**, **-t**, and **-f** options as shown below. This example initializes the MDS volume **video**:

   ```
   % mdsvolinit -s -t -f $ORACLE_HOME/vs30/admin/voltab video
   ```

   This command writes information from the **voltab** file to the MDS disks, this allows the media data store directory server (**mdsdirsrv**) to access the disks as an MDS volume set.

   > **CAUTION:** Using **-t**, which initializes a volume, erases all content from the specified volume. Once an MDS volume has been initialized, its properties (i.e. disk configuration, stripe width, raidsize) cannot be modified without destroying all the volume's content. If you wish to modify an MDS volume, first back up all of its content, so it can be reloaded into the modified volume.

   For more information on mdsvolinit, refer to Chapter 12, "Oracle Media Data Store (MDS) Utilities."

7. Execute the script **mnstop**. This stops all OMN components.

   ```
   % ./mnstop
   ```

For Further Information

For further information on tasks and procedures related to the MDS filesystem, refer to Chapter 7, “Oracle Media Data Store Tasks and Procedures”.

mdsdirsrv

mdsdirsrv (Media Data Store directory server**)** manages the layout of files in the MDS and grants access permission to requesting clients (such as **vspump**), to create, access, or modify files. **mdsdirsrv** determines if there is bandwidth available to allow access without losing the service quality to clients already accessing the MDS.

Syntax

```
mdsdirsrv [-h] [-V] [-T][ -b disk...] -f $ORACLE_HOME/vs30/admin/voltab
[-l volume-name...] [-w volume-name...] [-L ] [-W]
[-R resource-statement] [-P resource-file]
```

where:

-b if you want **mdsdirsrv** to access volumes containing failed disks, you must specify these disks with **-b**. Note that messages will be written to both the system console and the OVS log warning of the failed disks.

-f identifies the **voltab** file that defines the volume(s) to be accessed
by **mdsdirsrv**.

-h prints usage information.

-l mounts the specified volume in read-only mode.

-L mounts all volumes specified in the **voltab** file in read-only mode, except those volumes specified by **-w**.

-P specifies the resource file. See “Using Resource Descriptors” earlier in this chapter.

-R sets the specified resource descriptor. See “Using Resource Descriptors” earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-w mounts the specified volume in read-write mode

-W mounts all volumes specified in the **voltab** file in read-write mode, except those volumes specified by **-l**.

-V prints a version banner.

Read-Write Modes

MDS volumes can be mounted in the following modes:

Read-Write Mode

A volume mounted in read-write mode allows **mdsdirsrv** to both read and write to the volume.

Read-Only Mode

A volume mounted in read-only mode allows mdsdirsrv to read but not write to the volume.

Usage Notes

While starting, **mdsdirsrv** recognizes any failed disks in the MDS volume and writes messages identifying them to the console and the OVS log file. You can recreate data onto a new disk with the **mdsrebuild** utility. For information on **mdsrebuild**, refer to Chapter 12, “Oracle Media Data Store (MDS) Utilities”.

Only one **mdsdirsrv** process can be running at any one time.

Examples

Example 1 This example command starts **mdsdirsrv**, allowing it to read and write the MDS volumes specified in the voltab file located in the directory **$ORACLE_HOME/vs30/admin**:

```
% mdsdirsrv -W -f $ORACLE_HOME/vs30/admin/voltab
```

Example 2 This example mounts all volumes in read-write mode, except for the volume **training**, which is mounted in read-only mode:

```
% mdsdirsrv -W -l training -f $ORACLE_HOME/vs30/admin/voltab
```

Related Commands

mdsvolinit, mdsvolstat

mdsftpsrv

mdsftpsrv is an FTP server daemon which allows an FTP client to transfer files to and from the MDS.

Syntax

`mdsftpsrv [-a] [-s] [-u] [-g] [-w] [-h] [-V] [-T] [-A` *`alloc-size`*`] [-p` *`ftp-port`*`] [-R` *`resource-statement`*`] [-P` *`resource-file`*`]`
where:

-a allows a encrypted password to be specified. To use the password, the resource **mds.alternate** must be specified with **-R** and set to the crypted password. If **-a** is omitted, the password is taken from the system's standard authentication mechanism.

-A sets a default file creation size which is the maximum size of an MDS file FTP will create within the MDS. The **-A** option assumes that the **allo** command has not been specified by the FTP client, and overrides any file creation size sent by the FTP client.

-g on systems which support the notion of groups, restricts access to users belonging to the same group as the user who owns the **mdsftpsrv** process.

-h prints usage information.

-p specifies the port number on which to listen for incoming requests. Oracle recommends using port 1621.

-P specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.

-R sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.

-s on systems which support the notion of groups, restricts access to users belonging to the system group or equivalent.

-T enables verbose mode.

-u restricts access to the user who owns the **mdsftpsrv** process.

-w allows access by any authorized user.

-V prints a version banner.

Usage Notes

mdsftpsrv does not provide access to host file systems. To access files on host file systems, use the FTP daemon provided by the operating system or network vendor.

mdsftpsrv must listen on a different port than a server platform's existing FTP daemon (**ftpd**) to allow **mdsftpsrv** to coexist with **ftpd**. Oracle recommends port 1621 for this purpose.

Note: Since **mdsftpsrv** is assigned a port, the user must specify the port number as an additional argument to the FTP client when requesting a connection. For example:

```
% ftp oracle-sun 1621
Connected to oracle-sun.us.oracle.com.
```

As the MDS contains only binary data in the form of digitized video and audio files, only the transfer of binary data is supported. ASCII transfer mode is not supported.

mdsftpsrv can transfer data to either an FTP client or another FTP server (either **mdsftpsrv** or **ftpd**) using passive mode.

Examples

Example 1 This example starts **mdsftpsrv** with the **-g** option, restricting access to users belonging to the same group as the owner of the **mdsftpsrv** process. The **-p** specifies that **mdsftpsrv** listen for incoming requests on port 1621 the Oracle recommended value:

```
% mdsftpsrv -g -p 1621 &
```

Example 2 This example starts **mdsftpsrv** with **-w**, giving access to any authorized user on the system:

```
% mdsftpsrv -w -p 1621 &
```

Example 3 In this example, **mdsftpsrv** is started on a system port, which is any port numbered 0 to 1023. To start **mdsftpsrv** on a system port, you must be logged in as **root**. This example specifies that **mdsftpsrv** listen on system port 621 with the **-p** option, while restricting access to those users belonging to the same system group with **-s**:

```
# mdsftpsrv -s -p 621 &
```

Example 4 **mdsftpsrv** provides for additional security through the use of an encrypted password. You must first obtain the encrypted version of your UNIX account password:

ypmatch *username* passwd
where:

username
is your UNIX login ID.

For example, to obtain the encrypted password for user **sflyte**, enter **ypmatch sflyte passwd** at your shell prompt:

```
% ypmatch sflyte passwd
sflyte:iMMElWbxWACVk:20211:520:Sebastian Flyte,,,:/home/sflyte:/bin/csh
```

The second field of the returned output contains the encrypted password for user **sflyte**; in this instance **iMMElWbxWACVk**.

To specify the encrypted password as the MDS FTP login password, use the **-R** option to specify the **mds.alternate** resource. The syntax for the resource statement is:

mds.alternate=*password*
where:

password
is the encrypted output of **ypmatch**.

To use the encrypted password, use the **-a** option in conjunction with **-R** and the **mds.alternate** resource descriptor and password:

```
% mdsftpsrv -a -R mds.alternate=iMMElWbxWACVk
```

Note: The format of the encrypted password is platform specific.

Example 5 This example starts **mdsftpsrv** with **-A**, specifying a default **allo** size of 12 Megabytes:

```
mdsftpsrv -A 12000000
```

When using **mdsftpsrv** with **-A**, the default **allo** size should be set high enough to accommodate any anticipated file transfer. Sending the **allo** command for individual file transfers with the FTP client therefore becomes unnecessary. If you attempt to FTP a file whose size exceeds that set with **-A**, the file transfer will fail.

mdshsmsrv

mdshsmsrv (Hierarchical Storage Management server) manages the metadata pertaining to tertiary storage. For each tape, the **mdshsmsrv** metadata manages the following information:

- tape identifier
- associated volume
- tape thresholds
- reserved block on the tape
- error and usage statistics

Syntax

`mdshsmsrv [-h] [-V] [-T] [-f` *`init-file count`*`] [-t` *`init-tape count`*`] [-m` *`maximum-threads`*`] [-r mount-`*`retries`*`][-d device-name] [-R` *`resource-statement`*`] [-P` *`resource-file`*` ]` *`control-device volumes`*

where:

-d is the operating system-dependent device name for the tape stacker or robot.

-f creates a table capable of holding a specified number of files, overwriting any existing data. The **-f** flag is used only once, for HSM table initialization. You must create these data tables to use the HSM. When using **-f**, you must specify a number of files as an argument.

-h prints usage information.

-m specifies the maximum number of threads.

-P specifies the resource file. See “Using Resource Descriptors” earlier in this chapter.

-r specifies the number of attempts that should be made to mount a tape before abandoning the operation.

-R sets the specified resource descriptor. See “Using Resource Descriptors” earlier in this chapter.

-t	creates a table capable of holding a specified number of tapes, overwriting any existing data. The **-t** flag (like the -f flag), is used for HSM table initialization only. When using **-t**, you must specify a number of tapes as an argument.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
control device	name should be set to the name of the control device. For example, for the Ampex storage device enter “ampex”.
volumes	are the MDS volumes that have HSM support.

Usage Notes

mdshsmsrv issues commands to the robotic arm to load and unload tapes or CDs. The implementation of this interface is dependent on the particular tertiary storage device being used, and uses the storage device’s high-level control interface.

The **-f** and the **-t** flag are used only once, for HSM table initialization. When using **-f** or **-t**, you must specify an argument.

Examples

Example 1 This example creates an HSM data file for a stacker or robot (**/dev/amc0)** in the MDS volume **movies**:

```
mdshsmsrv -d /dev/amc0 -f 1000 -t 10 ampex movies
```

This command creates tables capable of holding 1000 files and 10 tapes, respectively. The **-d** option specifies the device name for the tape stacker, /dev/amc0. The control device is specified as **ampex**. Table-creation messages are stored in the log file, and this is where you should look to ensure that the tables were successfully created. (You can specify more than one volume at a time.)

WARNING: Once you have initialized these tables, do not run **mdshsmsrv** with the **-f** or **-t** options on these tables again. Doing so will overwrite all current information in the HSM database.

Example 2 This example starts the HSM processes for normal operation:

```
% mdshsmsrv -d /dev/amc0 ampex movies
```

mdsrmtsrv

mdsrmtsrv (MDS remote file server) enables remote access to the MDS, including an efficient interface for downloading BLOBs.

Syntax

```
mdsrmtsrv [-m] [-h] [-V] [-T] [-R resource-statement]
[-P resource-file ]
```

where:

-h prints usage information.

-m specifies the maximum number of threads in the Media Data Store remote file server. More threads enables more files to be accessed concurrently. The default value, which is used when you do not specify a **-m** value, is 8.

-P specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.

-R sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Usage Notes

For systems in which you expect many clients to access BLOBs concurrently or in which there exist many remote MDS clients, you should run multiple mdsrmtsrv processes. The number of processes you will need to start depends on your particular installation. In general, if performance is unsatisfactory, you should start up another **mdsrmtsrv** process, or increase the number of threads.

If you expect any MDS or video server utilities to be run remotely, you should start up **mdsrmtsrv**. For example, if you want users to be able to run **mdsdump** remotely, then **mdsrmtsrv**, which enables the remote IO to occur, must be started.

Example

This example initiates the MDS remote file server, using the default values of 8 for the maximum number of threads.

```
% mdsrmtsrv
```

This example is the same as the command issued in the ovsstart script shipped with the Oracle Video Server.

mdsxfrsrv

mdsxfrsrv transfers data between disks and tertiary storage. Generally, you should use one **mdsxfrsrv** process for each tertiary storage device in the system.

Syntax

```
mdsxfrsrv [-h] [-V] [-T] [-c cache-memory] [-d device-id]
[-r retries] [ -R resource-statement ] [ -P resource-file ]
device-name
```

where:

-c specifies the size of the memory cache (in bytes) to be used when the MDS is initialized. Increasing the cache size can sometimes provide better performance when copying files. When not specified **mdsxfrsrv** uses a cache sized based on the parameters of the MDS volume (the parameters specified in the **voltab** file). In general, specify 4 MB or more of memory cache to improve performance.

-d assigns the HSM device an internal (secondary) address that will be used for all tape-manipulator commands. The internal address is a string specific to the hardware device being used. For a discussion of supported tape drives refer to the *Oracle Video Server Administrator's Guide*.

-h prints usage information.

-P specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.

-r specifies the number of attempts that should be made to open a file before abandoning the operation. When not specified the default is four.

-R sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

device-name is the device name of the tape device.

Usage Notes

The **-c** (cache) flag is an important optional parameter to the **mdsxfrsrv** process. This flag enables you to set a larger memory cache when MDS is initialized. Generally, you will not need to alter the default value, but if you find that copying files takes longer than expected, increase this value to at least two times the largest tape block size of any tape.

Examples

Example 1 This example starts a single data transfer server for the tape device **/dev/rdst0.1.**

```
mdsxfrsrv -r 5 -d 100 /dev/rdst0.1 &
```

Example 2 This example defines a cache size of `2000000` bytes.

```
mdsxfrsrv -c 2000000 -r 5 -d 100 /dev/rst0.1
```

vsbcastsrv

vsbcastsrv (broadcast data service) manages broadcast data services for scheduling services. **vsbcastsrv** manages the schedule, exporter, channel and playout information stored in the database for **vsschdsrv** and **vsnvodsrv**.

Syntax

```
vsbcastsrv -c ${USERNAME}/${PASSWORD}@${ALIAS} [-h] [-n threads]
[-P resource-file] [-R resource-statement] [-T] [-V]
```

where:

-c database connection string. The format of which is:

```
username/password@alias
```

where:

- ***username***

  user account ID for the database storing logical content information

- ***password***

  password for the database account storing logical content information

- ***alias***

  TNS connect alias for your database installation. For a complete discussion of database authentication, refer to your Oracle database server documentation.

-h prints usage information.

-n specifies the number of threads to run process method requests. When not specified the default is 1.

-P specifies the resource file. See “Using Resource Descriptors” earlier in this chapter.

-R sets the specified resource descriptor. See “Using Resource Descriptors” earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Usage Notes

Since the scheduling service only plays logical content titles, the logical content service (initiated with **vscontsrv**) must be started prior to **vsbcastsrv**. In addition, logical content titles must be created in the database before scheduled playout services can be implemented. For information on starting the logical content service and creating logical content titles, refer to **Related Commands** and **For Further Information** later in this section.

Examples

Example 1 This example uses the **-c** *connect-string* option to authenticate a connection to the Oracle database. In this example the connect string authenticates the user **ovs** with password **ovs**.The file containing the database connect information is aliased as **ovsdb**:

```
% vsbcastsrv -c ovs/ovs@ovsdb
```

> **Note:** When using **-c** to connect to a database, the connect string information is not secure. Issuing a **ps** command will display the userid and password. If you require a secure database connection follow the steps in Example 2.

Example 2 To create a secure database connection, follow these steps:

1. If you are not using the Global Name Server, ensure that the database containing your logical content is defined in your **tnsnames.ora** file before initiating **vscontsrv** or **vsbcastsrv**.

   Here is a sample SQLnet connect descriptor from **tnsnames.ora:**

```
ovs=
          (DESCRIPTION =
                  (ADDRESS =
                          (PROTOCOL = TCP)
                          (Host = 199.25.102.201)
                          (Port = 1521))
(CONNECT_DATA = (SID = ovs)))
```

2. Edit the file **$ORACLE_HOME/vs30/admin/mnrc** and modify the connect string parameters for **vsbcastsrv.connect=**. The connect string is of the form:

   ```
   vsbcastsrv.connect=username/password@alias
   ```

   where:

username	user account ID for the database storing logical content information
password	password for the database account storing logical content information
alias	TNS connect alias for your database installation. For a complete discussion of database authentication, refer to your database server documentation.

The following example edits the file **$ORACLE_HOME/vs30/admin/mnrc** and adds the database connect string **ovs/ovs@ovs** to **vsbcastsrv**:

```
ys.log.msg-path=/home/oracle/vs30/mesg
ys.log.msg-path=/home/oracle/mn33/mesg
vscontsrv.connect=ovs/ovs@ovsdb
vsbcastsrv.connect=ovs/ovs@ovsdb
```

Related Commands

vsnvodsrv, **vscontsrv**, **vsschdsrv**

For Further Information

You will find information on:

- Scheduling Service and the creation of broadcast schedules in the *Oracle Video Server Developer's Guide*, and *Getting Started with Oracle Video Server Manager.*
- implementing the Scheduling Service in Chapter 9, "Configuring Logical Content and Scheduling Service".

vscontsrv

vscontsrv (content service) allows a client to query either the MDS file system or Oracle Video Server database and obtain a listing of titles. It also resolves titles on behalf of the stream service.

Syntax

vscontsrv [-c ${*USERNAME*}/${*PASSWORD*}@${*ALIAS*}] [-h] [-m *cursors*] [-n *threads*] [-P] [-R] [-T] [-V]
where:

-c database connection string. The format is:

`username/password@alias`

where:

- ***username***

  user account ID for the database storing logical content information

- ***password***

  password for the database account storing logical content information

- ***alias***

  TNS connect alias for your database installation. For a complete discussion of database authentication, refer to your Oracle database server documentation.

-h prints usage information.

-m **Note: At the time of this release, the -m option is no longer required. Do not use the -m option when configuring vscontsrv.**

specifies the maximum number of cursors a connection to the database may have open at a given time. When not specified the default is 20.

-n specifies the number of threads to run process method requests. When not specified the default is 1.

-P	specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.

Usage Notes

The content service can be used in either:

stand-alone mode
listing tag files contained in all mounted MDS volumes

database mode
listing logical content titles and clips in addition to listing tag files

The default content service is stand-alone mode. To use logical content, OVS must be installed with a connection to a separate Oracle database storing logical content information.

Specifying a Database

You can establish a database connection for **vscontsrv** in one of two ways:

- editing the **vscontsrv** entry in the file **$ORACLE_HOME/vs30/admin/ovsstart** to include the **-c *connect-string*** option
- editing the file **$ORACLE_HOME/vs30/admin/mnrc** to include **vscontsrv** with the resource descriptor **connect=*connect-string***

The preferred method is to edit the **mnrc** file. Placing database connection information in the **mnrc** file creates a secure database connection. Examples of both database connection methods are provided later in this section.

Configuring Threads

You can increase the number of concurrent database sessions **vscontsrv** can have with the Oracle database by increasing the number of threads. Modify this value with the option **-n *threads.*** When not specified the default value is 1 thread, allowing 3 concurrent database sessions.

For more information on determining values for cursors and threads, and how these values effect the number of concurrent database sessions, refer to Chapter 9, "Configuring the Logical Content and Scheduling Services."

Increasing Threads without a Database If you are operating Oracle Video Server without a database, increasing the number of threads can improve the performance of **vscontsrv**. You may wish to experiment with different thread values when operating Oracle Video Server in this way.

Examples

Example 1 This example starts the content service in stand-alone mode:

```
% vscontsrv &
```

To start Oracle Video Server without a databse connection (stand-alone mode) ensure that you:

- start **vscontsrv** without the **-c** option (see Example 2)
- remove the **vscontsrv.connect=*connect-string*** statement from the file **$ORACLE_HOME/vs30/admin/mnrc**.

If either of these conditions exists, Oracle Video Server will attempt to connect to the specified database.

Example 2 This example uses a logical content database. The **-c** *connect-string* option authenticates a connection to the database storing logical content information. In this example the connect string authenticates the user **oracle** with password **ovs**. The file containing the database connect information is aliased as **ovs**:

```
% vscontsrv -c oracle/ovs@ovs
```

> **Note:** When using **-c** to connect to a database, the connect string information is not secure. Issuing the Unix **ps** command will display the userid and password. If you require a secure database connection follow the steps in Example 3.

Example 3 To create a secure database connection, edit the file **$ORACLE_HOME/vs30/admin/mnrc** and modify the connect string parameters for **vscontsrv**. The following example edits the **mnrc** file and adds the database connect string **ovs/ovs@ovsdb** to **vscontsrv**:

```
ys.log.msg-path=/home/oracle/vs30/mesg
ys.log.msg-path=/home/oracle/mn33/mesg
vscontsrv.connect=ovs/ovs@ovsdb
vsnvodsrv.connect=ovs/ovs@ovsdb
```

The following entry is an example of a **tnsnames.ora** entry to add the database connect string to the database:

```
Sample db connect string from tnsnames.ora

ovs=
          (DESCRIPTION =
                  (ADDRESS =
                          (PROTOCOL = TCP)
                          (Host = 199.25.102.201)
                          (Port = 1521))
(CONNECT_DATA = (SID = ovs)))
```

Note: If you are running the Global Name Server you may not need this file or this entry. For a complete discussion of database authentication, refer to your Database Server documentation.

For Further Information

You will find information on:

- the logical content model and the creation of logical content titles in the *Oracle Video Server Manager Getting Started Guide.*
- implementing the Logical Content Service in Chapter 9, “Configuring the Logical Content and Scheduling Services”.

vscsmsrv

vscsmsrv (circuit and session manager) allocates sessions and circuits for client devices connecting to Oracle Video Server.

Syntax

`vscsmsrv [-f file:`*`filename`*`][-h] [-P `*`resource-file`*`] [-R `*`resource-statement`*`] [-T] [-V]`
where:

-f file:*filename*	specifies the configuration file **vscsmsrv** uses to allocate sessions and circuits. For information on constructing this file refer to Chapter 11, "Configuring the Session-and-Circuit Service."
-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.

Examples

Example 1 This example starts the default **vscsmsrv** service:

```
% vscsmsrv &
```

Example 2 This example specifies the session and circuit service configuration file **$ORACLE_HOME/vs30/admin/vcm.cfg**:

```
% vscsmsrv -f file:$ORACLE_HOME/vs30/admin/vcm.cfg
```

Related Commands

vscsmdir, vscsmkill

For Further Information

For further information on the Session-and-Circuit Service, refer to the following documents and sections:

- *Introducing Oracle Video Server*
- Chapter 10, "Configuring the Session-and-Circuit Service."

session-and-circuit configuration file

The session-and-circuit configuration file allows you to assign (or map) video pumps (or other channel providers) to specific *client groups*. A client group defines a group of clients belonging to a particular network segment. You can map client requests to video pumps to better deal with segmented network topologies. An advantage in creating such a configuration file is that it allows Oracle Video Server to use multiple network interface cards (NICs), providing increased network bandwidth and load balancing.

Syntax

<interface-information> + *<link-information>* + *<client-group>*
where:

interface-information	defines the server and the network interfaces available to Oracle Video Server.
link-information	defines paths of communication between the video pump and the network interfaces available to Oracle Video Server. The video pump uses communication links to deliver data to a client.
client-group	defines collections of clients and the video pumps that are compatible with them. Any video pump can service any client within its client group.

Each of these statements consists of several parameters. The following sections describe the syntax for each section of the configuration file:

Interface Information

```
<interface-info> = 'NIC' <interface-name> '{' <server-hostname>
                        <device-name> <maxbw> [<interface-info>] '}' ';'
```

where:

interface-name	string identifying the network interface.
server-hostname	hostname of the server running containing network interface card (NIC)
device-name	device name of the network interface card

maxbw sustained network bandwidth the network interface can provide. Generally, this is about 60–70% of the maximum throughput a network interface can produce.

interface-info additional information you may wish to note regarding the server or interface. **This information is optional**.

Communication links Information

```
<link-information> = 'LINK' <linkname> '{' <interface-name> <protocol>
                    <address> <pktpref> <pktmax> <pktmod> [<api-name>]
                    [<api-info>] '}' ';'
```

where:

linkname string identifying the link serving as a path of communication between the server and client.

interface-name name used to identify a network interface. The name used here must match a network device name used in the ***interface-info*** statement.

protocol network protocol used by the communications link. Typical protocols are UDP, TCP, and ATM (AAL5-SVC or AAL5-PVC).

address regular expression identifying client network addresses that can be serviced by this communication link.

pktpref preferred network packet size specified in bytes. The preferred packet size should match that specified with the **-n** option used by the video pump. In UDP environments this is typically 8192 bytes (8k).

pktmax maximum network packet size.

pktmod network packet size modulus. All packet sizes must be a multiple of this packet size.

api-name identifies the software API the video pump or other channel provider uses to access the network interface. To obtain information on the software API used by the network interface, refer to the documentation provided by the network card vendor. **This information is optional**.

api-info additional information about the network API. **This information is optional**.

Client Group Information

```
<client-group> = 'GROUP' <groupname> '{' 'PROVIDERS' '{' (<pvdrname>
                 <linkname> ';')+ '}' 'CLIENTS' '{' (<protocol-name>
                 <address> ';')+ '}' '}' ';'
```

where:

groupname string that identifies the group of clients serviced by a group of video pumps or other channel providers.

PROVIDERS identifies the section of the configuration file defining the channel providers (video pumps)

pvdr-name string identifying the video pumps or other channel providers.

linkname string that identifies the communications link serviced by a channel provider

protocol-name string identifying the network protocol to use in conjunction with a client group. Typical network protocols are UDP, TCP, and ATM (AAL5-SVC or AAL5-PVC).

address regular expression identifying network addresses of clients

Usage Notes

Using Regular Expressions to Define Client Addresses

Regular expressions define a set of one or more strings of characters. You can use a regular expression in the session-and-circuit configuration file to define specific network addresses of clients. The regular expression characters to specify network addresses of clients are described in Table 11-1.

Table 11–1 Regular expression characters

Expression	Meaning
C	The character ‘C’ where ‘C’ is not one of the following meta characters * ? [{
*****	Matches zero or more occurrences of the preceding character.
?	Matches zero or more occurrences of the proceeding character.
[xyz]	Matches any of the characters containing ‘x,’ ‘y,’ or ‘z.’
{RE1, RE2, RE3...}	Matches any of RE1, RE2, or, RE3.

Examples

For examples of session-and-circuit configuration files for specific network environments, see Chapter 10, “Configuring the Session-and-Circuit Service.”

Related Commands

vscsmsrv

For Further Information

For further information on the Session-and-Circuit Service, refer to the following documents and sections:

- *Introducing Oracle Video Server*
- Chapter 10, “Configuring the Session-and-Circuit Service”

vsfeedsrv

vsfeedsrv (real-time feed server) loads video data and metadata into the MDS directly from encoding stations in real-time. The real-time feed service supports one-step encoding and continuous real-time feeds.

Syntax

vsfeedsrv [-h] [-b max-session-bandwidth] [-m *min-content-time*][-s max-feed-session][-M *max-content-time*][-n *real-time feed files*] [-R *resource-statement*] [-P *resource-file*][T] [V]

-b specifies the maximum bandwidth of each feed. The default is 2048000 bits per second (2.048 Mbps)

-h prints usage information.

-m specifies the minimum file size (in time) to use when creating continuous feed files. Accepted modifier values are:

h (hours)

m (minutes)

s (seconds)

The default minimum file size is 60 seconds. If you do not specify a time interval (h, m, or s), seconds is used. Values lower than this are not recommended

-n specifies the target number of real-time feed files that the OVS attempts to create for each continuous feed. The default is 8 continuous feed files. Note that the actual number of feed files created may be modified as a result of the values used for **-m** and **-M**.

-s specifies the maximum number of real-time feed sessions the server will handle. The default is 2.

-M	specifies the maximum file size (in time) to use when creating continuous feed files. Accepted modifier values are: **h** (hours) **m** (minutes) **s** (seconds) The default maximum file size is 1800 seconds (30 minutes). If you do not specify a time interval (h, m, or s), seconds is used
-P	specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.

Usage Notes

vsfeedsrv implements the Real-time Feed service in Oracle Video Server. A real-time feed is constituted of video and/or audio data and its associated metadata. Encoded data and metadata is fed from an encoding station to the OVS, which automatically loads and tags the file into the MDS in one step.

The encoding station allows you to input the length of the feed (in hours, minutes, or seconds) destined for the server. **vsfeedsrv** divides the length of the feed by the number of files OVS can create for each feed. For example, if the encoder is programmed to accept ten hours of real-time video data, and **vsfeedsrv** creates the default of eight real-time feed files, each file will be one and a quarter hours in length:

$$\frac{10 \text{ hours of real-time content}}{8 \text{ real-time files}} = 1.25 \text{ hours of content per real-time file}$$

By setting the **-m** and **-M** options, a minimum and maximum size for files to be created can be specified. Note that the minimum and maximum file sizes might modify the number of real-time file created. For example, if the above value of eight real-time files is specified using ten hours of real-time feed content, each file created will be one and a quarter hours in length (see the above equation). If

however a maximum file size of 60 minutes is specified with **-n**, ten files will be created.

$$\frac{\textit{10 hours of real-time content}}{\textit{10 real-time files}} = \textit{1 hour (60 minutes) of content per real-time file}$$

By default, file sizes are specified in seconds. You can specify times in minutes or hours using the modifiers described earlier. Note that time measurements cannot be combined. For example, if you wish to specify a file size of one and a half minutes, you must specify 90 seconds, not 1 minute 30 seconds. Refer to the provided examples for more information.

OVS uses a root file name—specified by the encoder—to which **vsfeedsrv** appends an eight digit, monotonically ascending time stamp for each .mpg file created. For example, if the assigned MDS volume is **volume1**, and the root file name is **video**, **mdsdir** would list the following output:

```
% mdsdir -l /mds/volume1
Volume: /mds/volume1 8 matches
210m Jan 22 13:50:15 rw video01234567.mpg
210m Jan 22 14:20:15 rw video01234611.mpg
210m Jan 22 15:00:15 rw video01234885.mpg
...
27k Jan 22 13:50:15 rw video.mpi
```

Note that only one tag file is created for all files within a given real-time feed. The tag file uses the specified root file name with the extension **.mpi**. If another real-time feed session is fed into the same MDS volume which uses the same root file name as a previous real-time feed, the first tag file will be overwritten (but the content files will remain).

Examples

Example 1 This example starts **vsfeedsrv** with the default values for **-m** (60 seconds), **-M** (30 minutes), and **-n** (eight files):

```
% vsfeedsrv &
```

Example 2 In this example, **vsfeedsrv** attempts to create ten files on the server. The files created must be greater than five minutes but less than 30 minutes in length.

```
% vsfeedsrv -m 5m -M 30m -n 10
```

The parameters used for this example might be:

- The encoder sends a continuous feed of five hours in length to the OVS.

- **vsfeedsrv** creates ten real-time files (specified by **-n**).
- the maximum file size is 30 minutes (specified with **-M**)

This would produce ten 30 minute video files:

$$\frac{\textit{5 hours of real-time feed}}{\textit{10 real-time files}} = \textit{30 minutes of content per real-time file}$$

Example 3 In this example, **vsfeedsrv** attempts to create twenty files on the server. The files created must be greater than ten minutes but less than 60 minutes in length.

```
% vsfeedsrv -m 10m -M 60m -n 20
```

For Further Information

For further information on the real-time feed service, refer to the following documents and sections:

- Chapter 8, “Using Real-time Video Feeds”
- *Introducing Oracle Video Server*
- Chapter 7, “Extending Video Encoders for Real-time Feeds” in the *Oracle Video Server Developer’s Guide*
- Appendix G,“Setting Up OVS Sample Applications”, in the *Oracle Video Server Developer’s Guide*

The parameters used for this example might be:

- The encoder sends a continuous feed of twenty hours in length to OVS.
- **vsfeedsrv** creates twenty real-time feed files (specified by **-n**).

This would produce twenty 60 minute video files:

$$\frac{\textit{20 hours of real-time feed}}{\textit{20 real-time files}} = \textit{60 minutes of content per real-time file}$$

Example 4 In this example, **vsfeedsrv** is started to support 5 real time feed clients using the **-s** option.

```
% vsfeedsrv -s 5
```

vsnvodsrv

vsnvodsrv (exporter service) is one of three processes which makes up the Scheduling Service. The Scheduling Service allows you to schedule programs for broadcast on channels you specify. **vsnvodsrv** starts and stops playout of programs on specific channels.

Syntax

```
vsnvodsrv [-h] [-n threads] [-P resource-file] [-R resource-statement] [-T] [-V]
```

where:

-h prints usage information.

-n specifies the number of threads to run process method requests. When not specified the default is 3.

-P specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.

-R sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Usage Notes

The **vsnvodsrv** process is responsible for starting and stopping the playout of programs on network channels. To ensure proper performance, configure **vsnvodsrv** with an appropriate number of threads to run process method requests. In general, the number of threads should be equal to the number of concurrent program start times. For more information on configuring **vsnvodsrv**, refer to Chapter 9, "Configuring the Logical Content and Scheduling Services."

Example

Example 1 This example starts **vsnvodsrv** with the default value of three processing threads:

```
% vsnvodsrv &
```

Example 2 To specify additional threads to execute process method requests, use the **-n** option. The number of threads should be equal to the number of concurrent program starts. This example specifies that **vsnvodsrv** use five threads:

```
% vsnvodsrv -n 5 &
```

Related Commands

vsbcastsrv, **vscontsrv**, **vsschdsrv**

For Further Information

You will find information on:

- implementing the Scheduling Service and the scheduling of logical content through VSM in the *Getting Started with Oracle Video Server Manager.*
- enabling and configuring the Scheduling Service in Chapter 9, “Configuring the Logical Content and Scheduling Services”.

vspump

vspump (video pump) reads content files from the Oracle Media Data Store (MDS) and sends the requested data to clients over the network.

Syntax

```
vspump [-a] [-b integer] [-c integer] [-C] [-d] [-h] [-m integer]
[-n integer] [-N integer ] [-o label] [-q] [-P resource-file]
[-R resource-statement] [-t] [-T] [-V] [-y]
```

where:

-a wraps data in Oracle generic format.

Note: When streaming content to the Oracle Video Client (OVC), **-a** must be specified.

-b specifies the maximum data transmission rate per stream in bps per second. The default maximum is 3100000 bps (3.10 Mbps).

-c specifies the amount of memory in bytes per stream which hold data that **vspump** reads from the MDS before sending it.

During server startup, the **vspump** process queries both the operating system and the MDS file system to ascertain how many bytes of data to hold in memory. Occasionally, it may be necessary to specify the value in the vspump configuration to optimize system performance. Note that specifying this option will use additional memory, and may create delays when clients request playback of a file.

-h prints usage information.

-m specifies the maximum number of streams **vspump** can serve.

-n sets the size in bytes of the output buffer. When using Oracle Video Client (OVC) in a LAN environment, the packet size should be 8192 bytes. When streaming low bit rate content, specify a packet size of 1024 bytes.

-N	specifies the number of packets **vspump** transmits at a time. The default is one and the maximum is four.
-o	specifies a reference label for an instance of **vspump** (i.e., **vspump1**). The reference label is used by the Session-and-Circuit Service to assign resources to a specific **vspump** instance. The label you assign with **-o** is the channel-provider name to use when creating a session-and-circuit configuration file. For more information on this refer to Chapter 10, "Configuring the Session-and-Circuit Service."
-P	specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-t	specifies that **vspump** operate as a real-time client to the MDS on Sun SPARC Solaris platforms. This option should always be used on Solaris platforms to specify real-time scheduling and to lock down the memory required for **vspump**.
-V	prints a version banner.
-y	allows an instance of **vspump** to bind to a CPU for its exclusive use. This allows the video pump to make better scheduling decisions, since it won't need to share the CPU with other processes. When operating the OVS on multi-processor platform, you should specify the **-y** option **Note: Do not specify this option on single-processor servers.**

Option Used for Interactive Television (ITV) Deployments

The following options are used in Interactive Television deployments.

Note: Do not use the following options when using Oracle Video Server in an IP network environment. These options are intended for use in deployments which include interactive television, and do not apply to video supplied over a local-area network.

-C transmits data without wrapping it in either Oracle generic or MPEG-2 format.

-d allows you to operate **vspump** without sending video data to a client for testing purposes.

Example

Example 1 The following example starts **vspump** with five concurrent streams specified with **-m** and a maximum bit rate of 3100000 bps specified with **-b**:

```
% vspump -a -S -o vspump1 -m 5 -n 8192 -b 3100000 -t &
```

Example 2 This example starts two instances of vspump (**vspump1** and **vspump2**).

This example uses the **-y** option to specify that one **vspump** be started on processor 2 and another on processor 3. The **-m** option specifies that each **vspump** instance supports 10 concurrent streams using a maximum bit rate of **3100000** bps (specified with the **-b** option). Note that the buffer size specified with **-n** is set to 8192 bytes (8K).

Note: When assigning multiple instances of **vspump**, each **vspump** instance must be assigned to its own processor. Ensure that each **vspump** has been assigned root privileges before assigning processors with the **-y** option. For information on assigning root privilege to **vspump**, refer to Chapter 5, "Post-Installation Tasks for the Oracle Video Server" in the *Oracle Video Server Installation Guide*.

```
% vspump -a -o vspump1 -m 10 -n 8192 -b 3100000 -y 2 -t &
% vspump -a -o vspump2 -m 10 -n 8192 -b 3100000 -y 3 -t &
```

Example 3 This example configures two video pumps. Note that the buffer size is specified with **-n**:

- **vspump1** delivers video to a high bit rate client using a packet size of 8K (8192)

- **vspump2** delivers video to a low bit rate client using a packet size of 1K (1024)

```
% vspump -a -o vspump1 -m 10 -n 8192 -b 3100000 -y 2 -t
% vspump -a -o vspump2 -m 10 -n 1024 -b 3100000 -y 3 -t
```

vsschdsrv

vsschdsrv (scheduling service) is a generic scheduling service that reads scheduling information and keeps track of time. Working with **vsbcastsrv** and **vsnvodsrv**, it initiates the playout of broadcast events at the scheduled time.

Syntax

```
vsschdsrv [-hPRTV] [-t wake-up] [-n threads] [-R resource statement]
[-P resource file][-T][-V]
```

where:

-h prints usage information.

-n specifies the number of service threads to concurrently run. Service threads handle events to play logical content titles at scheduled times and client status requests. When not specified the default is 1 service thread.

-t specifies how frequently **vsschdsrv** reads new schedule items from the broadcast data service (implemented by **vsbcastsrv**) in seconds. The time specified also determines how much of an interval will be read from **vsbcastsrv**. When not specified the default is 1 minute.

-P specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.

-R sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Usage Notes

Before starting **vsschdsrv**, you must create a logical content database and start both **vscontsrv** (content service) and **vsbcastsrv** (broadcast data service) using a database connection. Additionally, if you are not using the Global Name Server, you should make sure that the database containing your logical content is defined in the **tnsnames.ora** file.

Creating Scheduled Playout

To create scheduled playout, use Video Server Manager (VSM). For information on this and other VSM tasks, refer to the document *Getting Started with Oracle Video Server Manager.*

Examples

Example 1 This example starts the default **vsschdsrv** service:

```
% vsschdsrv &
```

Example 2 This example specifies that **vsschdsrv** read program schedules every 30 seconds by specifying **-t 30**:

```
% vsschdsrv -t 30 &
```

Related Commands

vsnvodsrv, **vsbcastsrv, vscontsrv**

For Further Information

You will find information on:

- Scheduling services and the scheduling of playout in the *Oracle Video Server Developer's Guide* and in the *Getting Started with Oracle Video Server Manager.*
- Implementing scheduled playout services, also known as NVOD, in Chapter 9, "Configuring Logical Content and Scheduling Services".

vsstrmsrv

vsstrmsrv (stream service) handles requests for streamed content from a client. When **vsstrmsrv** receives a request for a title, it resolves that title to one or more physical segments to be played. It then communicates with **vspump** to deliver those segments to the client in real time.

Syntax

`vsstrmsrv [-f] [-h] [-n integer] [-p] [-P] [-R] [-T] [-V]`
where:

-h prints usage information.

-n maximum number of clients which can concurrently connect to the stream service. The default is 20.

-P specifies the resource file. See "Using Resource Descriptors" earlier in this chapter.

-R sets the specified resource descriptor. See "Using Resource Descriptors" earlier in this chapter.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Example

To start **vsstrmsrv**, use this syntax:

```
% vsstrmsrv &
```

12

Oracle Media Data Store (MDS) Utilities

The Oracle Media Data Store (MDS) utilities perform standard file operations for the MDS, the RAID-based real-time file system for the Oracle Video Server (OVS). The MDS utilities are:

mdschecksum	Calculates and prints a checksum for any MDS or host file. The checksum is a 16-bit value expressed in hexadecimal.
mdsconcat	Concatenates files on either the host computer or within an MDS volume.
mdscopy	Copies one file to another or a number of files to an MDS volume or to a directory on the host.
mdsdefrag	Defragments MDS volumes.
mdsdelete	Removes files from MDS volumes.
mdsdir	Provides information about an MDS volume and the files it contains.
mdsdiskmode	Places disks into either rebuild mode or normal mode. You must put a disk into rebuild mode before rebuilding its data with the **mdsrebuild** utility. **mdsdiskmode** is also used to spare and unspare a disk.
mdsdump	Displays the hexadecimal equivalent of a file in an MDS volume or on a host file system.
mdshsmctl	Controls the registration of tapes and files with the HSM server and enables operator-initiated backup and restoration of (MDS) files.

mdshsmdir	Lists the files and tapes that have been registered with HSM, and the metadata associated with each one.
mdslock	Locks a file in an MDS volume in read-only mode. When a file is locked in read-only mode, it can be read but not written.
mdsrebuild	Rebuilds data onto a new disk after a disk failure.
mdsrename	Renames an existing file in an MDS volume.
mdstar	Loads files from an MDS volume or host to a tape drive or from a tape drive to an MDS volume or host.
mdsundelete	Restores a file that has been previously removed from the MDS volume with the **mdsdelete** utility.
mdsunlock	Unlocks a file in the MDS volume that was locked in read-only mode with the **mdslock** utility.
mdsvolinit	Initializes all disks of an MDS volumes with information from the **voltab** file.
mdsvolstat	Provides general information and statistics about MDS volumes.

mdschecksum

mdschecksum calculates and prints a checksum for any MDS or host file. The checksum is a 16-bit value expressed in hexadecimal notation.

Syntax

```
mdschecksum [-h] [-V] [-T] [-e {integer |0xinteger}] [-l {integer | 0xinteger}
[-s {integer |0xinteger}] [-R resource-statement]
[-P resource-file] filename...
```

where:

e checksums the file stopping with the specified decimal or hexadecimal byte position. If you omit **-e**, **mdschecksum** continues to the end of the file.

-h prints usage information.

-l length of the file to checksum in bytes.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s checksums the file starting with the specified decimal or hexadecimal byte position. If you omit **-s**, **mdschecksum** starts at the beginning of the file.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

filename the specified file.You can repeat this syntax to checksum several files.

Usage Notes

You can use **mdschecksum** to determine whether files were corrupted by large file operations such as moving several files from file system to file system. Run **mdschecksum**, record the checksum values, perform the file operations, run **mdschecksum** again, and compare the checksum values with those you calculated originally.

Example

This example calculates and prints the checksum for the file **oracle1.mpg** located in the MDS volume **video**:

```
% mdschecksum /mds/video/oracle1.mpg
oracle1.mpg:  cf5f
```

The checksum value is **cf5f**.

Related Commands

vsmpegchk

mdsconcat

mdsconcat concatenates MDS or host files together. If you have a content file split across multiple files you may want to concatenate them so that the content is in a single file.

Syntax

```
mdsconcat [-h] [-V] [-T] [-R resource-statement] [-P resource-file]
input_filename1 input_filename2... output_filename
```

where:

-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
input_filename	specifies two or more files to be concatenated.
output_filename	specifies the result of the concatenation.

Usage Notes

mdsconcat requires at least two input files.

Although **mdsconcat** will accept any input files, you should concatenate only content files that originated from a single file. Do not concatenate files from different sources. For example, do not combine a commercial and a movie file into one file with this utility.

Although you can concatenate content files, you cannot concatenate the individual tag files to create a single tag file for the resulting content file. You must generate a tag file for the concatenated file with the **vstag** utility.

Example

This example concatenates **video_clip1.mpg** and **video_clip2.mpg** into **final_video.mpg**:

```
% mdsconcat /mds/video/video_clip1.mpg /mds/video/video_clip2.mpg
/mds/video/final_video.mpg
```

mdscopy

mdscopy copies one file to another, or a number of files to an MDS volume or a directory on a host file system.

Syntax

Copying One File This syntax copies one file to another:

```
mdscopy [-h] [-T] [-V] [-s {integer | 0xinteger}]
[-l {integer | 0xinteger}] [-e {integer | 0xinteger}] [-p integer]
[-n slave-specification] [-R resource-statement] [-P resource-file] filename1
filename2
```

where:

-e copies the file stopping with the specified decimal or hexadecimal byte position. If you omit **-e**, mdscopy copies to the end of the file.

-h prints usage information.

-l specifies length of the file to copy in bytes.

-n runs parallel copying processes in the configuration specified. This is platform specific, generally applying to massively parallel hardware platforms.

-p specifies the number of parallel copying processes. If you omit -p, mdscopy does not start parallel copying processes and performs all the copying itself.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s copies the file starting with the specified decimal or hexadecimal byte position. If you omit -s, mdscopy copies from the beginning of the file.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

filename1 is the source file to be copied

filename2 is the destination file. If this file already exists, mdscopy overwrites it with the contents of the source file. If the source file is in the MDS and you include the volume or host file system directory but omit the filename, **mdscopy** will use the source file's filename.

Copying Several Files This syntax copies a number of files to an MDS volume or a directory on a host file system:

```
mdscopy [ -s {integer | 0xinteger} ] [ -e {integer | 0xinteger} ]
[-p integer ] filename... volume-name
```

where:

-s, **-e**, and **-p** are the same as in the previous syntax and apply to each file individually.

filename
is one or more source files to be copied.

volume-name
is the destination to which the source files are copied. The destination directory must already exist.

Usage Notes

Use the **-p** option of **mdscopy** to copy files using parallel copying processes. On some systems copying a file in parallel is faster than copying it with only one process. Since each file is copied by all parallel processes, parallel copying improves performance whether you copy several files or only one.

Note: It is recommended that parallel processes *only* be used on systems where MPP servers are being used.

Examples

Example 1 This example copies the file **oracle1.mpg** from the MDS volume **video** to the volume **video2**:

```
% mdscopy /mds/video/oracle1.mpg /mds/video2/oracle1.mpg
```

Example 2 This example uses a wildcard search to copy all of the files in the MDS volume **video** to the MDS volume **video2**. Note the use of quotes on the volume name. When using a wildcard search on an MDS volume, the full volume name must be specified in the search string When specifying a wildcard search within an MDS volume, use an asterisk (*) to specify files:

```
% mdscopy "/mds/video/*" /mds/video2
```

Example 3 This example uses a wildcard search to copy all files ending in **.mpg** from the current host directory into the MDS volume **video**:

```
% mdscopy ./*mpi /mds/video
```

Example 4 This example uses a wildcard search to copy all files in the MDS volume **movies** to the /usr/backup directory on the video server host:

```
% mdscopy "/mds/movies/*" /usr/backup
```

Related Commands

mdsdelete, **mdsdir**, **mdsundelete**

mdsdefrag

mdsdefrag defragments MDS volumes to preserve storage capacity.

Syntax

```
mdsdefrag volume_name...
```

where:

volume_name specifies the MDS volumes to be defragmented. Note that volumes must be mounted in read-write mode to be defragmented.

Usage Notes

The **mdsdefrag** utility can be used while the Oracle Video Server is streaming video or while content is being added to or deleted from an MDS volume. Please note that **mdsdefrag** may wait to perform file operations if files are being added to or deleted from a volume, or if a volume is being heavily accessed by MDS clients. If **mdsdefrag** is interrupted by other file operations it creates a file, called **MDSDFRAG-TMP**, which is removed when defragmenting is resumed.

> **Note:** During defragmenting, do not remove the **MDSDFRAG-TMP** file. **mdsdefrag** will remove the file when it completes the defragmenting process. If **mdsdefrag** is stopped prior to removing **MDSDFRAG-TMP, mdsdefrag** will remove it when it is next run.

If you omit a list of volume names, **mdsdefrag** will default to all volumes mounted in read-write mode.

To monitor the operation of **mdsdefrag**, use the **mnlogreader** process to read the server generated log messages; **mdsdefrag** does not write to the console.

The **mdsdefrag** utility is selective when moving files, and tries to minimize the amount of data moved. If the amount of data to be relocated is very large in relation to the perceived benefit, **mdsdefrag** may not defragment the volume completely.

mdsdefrag can defragment read-only and locked files.

Example This example defragments the MDS volume **video**:

```
% mdsdefrag video
```

For Further Information

You will find more information about **mdsdefrag** in Chapter 7, “Oracle Media Data Store Tasks and Procedures”.

mdsdelete

mdsdelete removes files from the MDS file system.

Syntax

```
mdsdelete [-h] [-V] [-T] [-e] [-z] [-R resource-descriptor]
[-P resource-file] mds-filename...
```

where:

-e	clears a file from the TOC of the MDS making it invisible to OVS components. If a file is removed using the **-e** option it cannot be recovered with mdsundelete.
-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
-z	clears a file from the TOC of the MDS, making it invisible to OVS components, and overwrites the disk space with zeroes. If a file is removed using the **-z** option it cannot be recovered with mdsundelete.
mds-filename	specifies one or more files to be removed.

Usage Notes

If you inadvertently remove a file from an MDS volume, you can restore the file with the **mdsundelete** utility any time until the file is overwritten by another file loaded into the same volume. You can determine whether the file has been overwritten with **mdsdir**. The overwriting of invisible, removed files by newly loaded files is not predictable.

You cannot restore a file that you remove with the **-e** or **-z** option. Files removed with these options are unrecoverable.

Examples

Example 1 This example deletes the file **oracle1.mpg** from the volume **video**:

```
% mdsdelete /mds/video/oracle1.mpg
```

> **Note:** Since this example does not use the **-z** or **-e** flags, you can still restore this file if it hasn't been overwritten.

Example 2 You can remove several files at once using the asterisk (*) wildcard character. This example removes all files in the MDS volume **video** ending with the extension **.mpg**:

```
% mdsdelete "/mds/video/*.mpg"
```

Note that to use a wildcard regular expression you must enclose the search string in double quotes. For example, the string in this example searches for all files ending with the extension **.mpg** contained in the MDS volume **video**.

Example 3 This example uses the **-z** option to erase the file **video1.osf** from the volume **video**:

```
% mdsdelete -z /mds/video/video1.osf
```

The file **video1**.osf cannot be restored with **mdsundelete**.

Example 4 In this example, the file **oracle1.mpg** is made invisible to Oracle Video Server components with the **-e** option:

```
% mdsdelete -e /mds/video/oracle1.mpg
```

When a file is made invisible to Oracle Video Server components, it cannot be retrieved by a client or manipulated with Oracle Video Server file utilities.

Related Commands

mdsundelete

mdsdir

mdsdir provides information about MDS volumes and the files they contain.

Syntax

```
mdsdir [-a] [-b] [-v] [-d] [-f] [-l] [-p] [-s] [-h] [-V] [-T]
[-R resource-statement] [-P resource-file] [volume-name...]
```

where:

-a	lists all files, including files that have been removed but not yet overwritten, in the specified volume. This option can be used with other **mdsdir** options.
-b	lists filenames, dates, and sizes in bytes.
-d	prints out the status of each disk in a volume: normal, rebuild, and / or if it is spared.
-f	lists free space in the specified volume.
-h	prints usage information.
-l	lists file names, sizes, dates, and whether each file is locked.
-p	lists a physical map of the specified volume.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-s	displays only the volume names.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-v	lists the statistics of the specified volume.
-V	prints version banner. If no options are specified, **mdsdir** lists only filenames.
volume-name	lists the files in the specified *volume*. If you do not specify a volume, **mdsdir** lists files in all MDS volumes.

Examples

Example 1 This example lists the files in the volume **/mds/video**, including files that have been deleted but not yet overwritten:

```
% mdsdir -a /mds/video
```

The output might look like this:

```
Volume:  /mds/video (ro)   12 matches
acntrl0.txt     clapton2.mpg    clifprev.mpg    cliftitl.mpg
control0.txt    dls14.mpg       dls28.mpg       mason.mpg
mason.txt       masont2.mpg     mdsdump.test    pvrintro.mpf(DEL)
```

This listing shows:

- The (ro) after **/mds/video** indicates the volume video is mounted in read-only mode:
  - An (rw) would indicate the volume is mounted in read-write mode.
- The video volume contains 12 files, ten of which are valid and accessible.
- The file pvrintro.mpf has been deleted but not overwritten. If you inadvertently remove a file, you can restore it with the **mdsundelete** utility.

Example 2 This example lists the free space (**-f**) in the MDS volume **video**:

```
% mdsdir -f /mds/video
Volume:  /mds/video 2 matches
1985 MB's of free space in 1 fragment(s)
Largest free block is 1985 MB's
```

Example 3 This example uses the **-l** option to list files in the volume /mds/video with the size, date and time, read/write mode, and name of each file:

```
% mdsdir -l /mds/video
Volume:  /mds/video 50 matches
 294m  Feb 26 17:09 rw oracle1.mpg
 273b  Feb 26 17:19 rw oracle1.mpi
  19m  Feb 26 16:57 rw oracle2.mpg
  94m  Feb 26 16:57 r- oracle3.mpg
  10m  Feb 26 16:57 rw oracle4.mpg
 172b  Mar 18 00:04 r- oracle4.mpi
  10m  Feb 26 16:57 rw oracle5.mpg
```

The **-l** listing uses:

- the letters **b** (bytes), **k** (kilobytes), and **m** (megabytes) to express file sizes
- the letter **r** to denote files locked in read-only mode and the letters rw to denote those in read-write mode

Example 4 This example also lists files in the volume /mds/video with the size (in bytes), date and time, read/write mode, and name of each file:

```
% mdsdir -b /mds/video2
Volume:  /mds/video2 50 matches
 308258300  Feb 26 17:09 rw oracle1.mpg
 273        Feb 26 17:19 rw oracle1.mpi
 308258300  Feb 26 16:57 rw oracle2.mpg
 926016     Feb 26 16:57 r- oracle3.mpg
 20441688   Feb 26 16:57 rw oracle4.mpg
...
```

The **-b** listing gives the exact size of each file in bytes.

Example 5 This example gives a physical map (**-p**) of the volume /mds/video2:

```
% mdsdir -p /mds/video2
Volume:  /mds/video2 30 matches
    1   start   6182 len  405091608    Feb 26 16:57   oracle1.mpg
 6183   start     15 len     926016    Feb 26 16:57   oracle2.mpg
 6198   start    160 len   10467980    Feb 26 16:57   dls14.mpg
 6358   start     43 len    2801408    Feb 26 16:57   dls28.mpg
12509   start   1836 len  120324096 ------------------Free Space
14345   start    288 len   18830832    Feb 26 16:47   vv13.mpg
64579   start   1577 len   10335027 ------------------Free Space
1034 MB's of free space in 4 fragment(s)
Largest free block is 782 MB's
```

The file dls28.mpg, for example, begins in RAID stripe 6358, occupies 43 RAID stripes, and is 2801408 bytes long.

Related Commands

mdscopy, **mdsdelete**

mdsdiskmode

mdsdiskmode puts disks into either rebuild mode or normal mode. You must put a disk into rebuild mode before:

- rebuilding its data with the **mdsrebuild** utility
- installing a new disk
- removing a failed disk

mdsdiskmode is also used to replace a failed disks with a spare.

For more information on transferring data to a spare disk refer to Chapter 7, "Oracle Media Data Store Tasks and Procedures".

Syntax

```
mdsdiskmode [-h] [-V] [-T] -f voltab {-r|-n|-o|-s spare-disk-name|-u}
[-R resource-statement] [-P resource-file] disk
```

where:

-f identifies the **voltab** file containing the definition for the volume containing the disk, such as $ORACLE_HOME/vs30/admin/voltab.

-h prints usage information.

-n puts disks into normal mode. In normal mode a disk can be both written and read.

-o puts disks into off-line mode. In off-line mode a disk cannot be written or read.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-r puts disks into rebuild mode. In rebuild mode a disk can be written but not read.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s spares a failed disk to the designated spare disk and places it in rebuild mode.

-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-u	unspares a spare disk and transfers disk I/O operations to the replaced disk in the specified volume.
-V	prints a version banner.
disk	specifies the disk to put in rebuild or normal mode.

Usage Notes

After rebuilding data onto a new disk with the **mdsrebuild** utility, do not use **mdsdiskmode** to put the disk into normal mode. **mdsrebuild** automatically puts the disk into normal mode after rebuilding data onto it.

When **mdsdirsrv** starts, it puts each disk in each of its volumes into the mode that the disk was in when **mdsdirsrv** was last shut down. For example, if the system crashes after you place a new disk into rebuild mode with **mdsdiskmode**, and then began rebuilding data onto that disk with **mdsrebuild**, restarting **mdsdirsrv** places the disk in rebuild mode. You should then restart rebuilding data onto the failed disk.

If you have specified a spare disk in your volume specification, **mdsdiskmode** can be used to put a spare disk in rebuild mode using the **-s** option. This prepares the spare disk to be rebuilt from the other disks in the MDS volume. Note that you can only use the **-s** and **-u** options for MDS volumes where a spare disk has been specified.

Unlike other Oracle Video Server utilities, **mdsdiskmode** can be executed when the server is up or down.

The fact that you are using **mdsdiskmode** probably means one or more of your drives is experiencing a hardware failure. **mdsdiskmode** may take quite some time (10 seconds or more) to initialize in this case.

Examples

Example 1 This example puts the disk /dev/rdsk/c0t1d0s6 into rebuild mode:

```
% mdsdiskmode -f $ORACLE_HOME/vs30/admin/voltab -r /dev/rdsk/c0t1d0s6
```

Example 2 This example spares the failed disk /dev/rdsk/c0t1d0s0 to the spare disk /dev/rdsk/c3t5d0s0:

```
% mdsdiskmode -f $ORACLE_HOME/vs30/admin/voltab -s /dev/rdsk/c3t5d0s0
/dev/rdsk/c0t1d0s0 .
```

For a detailed example of replacing a failed disk with a spare, refer to Chapter 5, “Oracle Media Data Store Tasks and Procedures”.

Related Commands

mdsrebuild

mdsdump

mdsdump shows you the hexadecimal equivalent of a file in the MDS or host file system.

Syntax

```
mdsdump [-h] [-V] [-T] [-s {integer | 0xinteger}]
[-e {integer|0xinteger}|-l {integer|0xinteger}] [-R resource-descriptor]
[-P resource-file] filename...
```

where:

-e dumps each file stopping with the specified decimal or hexadecimal byte position. If you omit -e, mdsdump dumps each file to its end.

-h prints usage information

-l length of the file to dump in bytes.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s dumps each file starting with the specified decimal or hexadecimal byte position. If you omit **-s**, mdsdump dumps each file from its beginning.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

filename specifies one or more files to be dumped.

Usage Notes

If you dump multiple files, the output for each file is preceded by the file name.

The output of **mdsdump** might look like this:

```
0000000020 6778 4669 6c65 6e61 6d65 4669 6c74 6572 gxFilenameFilter
0000000030 2074 6861 7420 7061 7273 6573 2061 2073  that parses a s
0000000040 7472 696e 6720 696e 746f 0a61 2076 6f6c tring into.a vol
0000000050 756d 656e 616d 6520 616e 6420 6120 6669 umename and a fi
```

```
0000000060 6c65 6e61 6d65 2e0a 0a52 756c 6520 313a lename...Rule 1:
0000000070 2076 6f6c 756d 6520 6f72 2066 696c 6520  volume or file
0000000080 6e61 6d65 7320 646f 6e27 7420 636f 6e74 names don't cont
0000000090 6169 6e20 2f2e 0a52 756c 6520 323a 206e ain /..Rule 2: n
00000000a0 616d 6573 2061 7265 2063 6173 6520 696e ames are case in
00000000b0 7365 6e73 6974 6976 652e 2020 464f 4f20 sensitive.  FOO
00000000c0 3d20 666f 6f2e 0a52 756c 6520 333a 2074 = foo..Rule 3: t
00000000d0 6865 2061 6273 6f6c 7574 6520 7061 7468 he absolute path
00000000e0 2066 6f72 2061 6e79 2066 696c 6520 6973  for any file is
```

Output is arranged in multiple rows. Each row contains 10 fields and represents 16 bytes of data:

- The first field has 10 hexadecimal digits and is the address of the 16 bytes in the file.
- The next 8 fields each have 4 hexadecimal digits. Each represents 2 bytes of data (2 digits represent a byte). These bytes appear in order and are not byte swapped.
- The last field is a 16 character representation of the 16 bytes. An unprintable character is represented by a period (.).

To save space, repeated rows are not printed. An asterisk (*) represents one or more rows identical to the previous row.

```
0000000000 2321 2f62 696e 2f73 680a 6578 6563 204e #!/bin/sh.exec N
0000000010 4320 2430 2022 2440 220a 0000 0000 0000 C $0 "$@".......
0000000020 2d6e 2031 0a00 0000 0000 0000 0000 0000 -n 1............
0000000030 0000 0000 0000 0000 0000 0000 0000 0000 ................
*
0000000100 7f45 4c46 0101 0100 0000 0000 0000 0000 ELF............
0000000110 0200 1000 0100 0000 0000 0000 3400 0000 ............4...
0000000120 40d7 0700 0000 0000 3400 2000 0300 2800 @.......4. ...(.
0000000130 0c00 0800 0600 0000 9800 0000 0000 0000 ................
0000000140 0000 0000 4000 0000 4000 0000 0400 0000 ....@...@.......
0000000150 0000 0000 0100 0000 d800 0000 0000 0000 ................
0000000160 0300 0000 4c14 0300 4c14 0300 0500 0000 ....L...L.......
```

Example

This example returns the hexadecimal equivalent of the file **test1** from the address 0x20 to 0x100, as in the first example in the "Usage Notes"section:

```
% mdsdump -s 0x20 -e 0x100 /mds/video/test1
```

Related Commands

mdschecksum, **vsmpegchk**

mdshsmctl

mdshsmctl controls the registration of tapes and files with the HSM server. It also enables operator-initiated backup and restoration of (MDS) files.

Syntax

```
mdshsmctl [-b] [-c] [-d] [-r] [-f] [-t] [-h] [-V] [-T] [-B integer]
[-v volume] [-l integer] [-R resource-statement] [-P resource-file]
file-or-tapes...
```

where:

-b	backs up a registered file to tape.
-B	specifies the block size to use. Ensure that this matches the block size of the tape device being used. Used with the **-c** and **-t** flag only.
-c	creates a tape or file.
-d	deletes a tape or file from the HSM.
-f	specifies a file (not a tape). Used in conjunction with the **-c** and **-d** flags.
-h	prints usage information.
-l	specifies the length of the tape in blocks. Used with the **-c** and **-t** flags.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-r	restores a file from tape to the MDS volume.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-t	specifies a tape (not a file). Used in conjunction with the **-c** and **-d** flags.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-v	specifies the MDS volume to which the tape or file is associated.
-V	prints a version banner.

Usage Notes

Block Sizes Some tape devices may have special restrictions on block sizes. Be sure to check for these restrictions before specifying the block size. For more information on block sizes, refer to "Choosing a Tape Block Size" in Chapter 7, "Oracle Media Data Store Tasks and Procedures".

Specifying Length of Tape Be conservative when specifying the length of a tape (**-l** flag); if you specify too long a length, you risk running out of room on the tape itself, resulting in a wasted transfer and additional system administration.

Registering Tapes with HSM You register tapes only once—when they are introduced to the system. Also, you may use a `k` or `m` suffix when allocating size, to indicate whether the allocation should be measured in kilo (1024) or mega (1048576) units. Case is ignored; you specify **k** or **K**, **m** or **M**.

Examples

Example 1 This example registers two tapes to be loaded on the system, by creating a record in the HSM database for each tape:

```
% mdshsmctl -ct -v movies -B lm -1 160k 000001
% mdshsmctl -ct -v movies -B lm -1 160k 000002
```

> **Note:** This registration process only creates a *description* of the tape; it does not actually verify that such a tape is present on the system.

Example 2 This example registers the file oracle1.mpg with tape **000001** on the HSM system:

```
% mdshsmctl -cf oracle1.mpg 000001
```

Example 3 This example performs a copy / backup on the file oracle1.mpg:

```
% mdshsmctl -b oracle1.mpg
```

Example 4 This example restores the file oracle1.mpg after it has been deleted from the MDS:

```
% mdshsmctl -r oracle1.mpg
```

Example 5 This example deletes a tape (and its contents) from the HSM database:

```
% mdshsmctl -dt 000001
```

> **Note:** Before a tape can be deleted from the HSM database, you must first delete all the files which were on the tape from the database. This safety measure ensures that tapes containing active content are not deleted from the database.

Related Commands

mdshsmdir

mdshsmdir

mdshsmdir lists the files and tapes that have been registered with HSM, and the metadata associated with each.

Syntax

```
mdshsmdir [-t] [-s] [-h] [-V] [-T] [-R resource-statement]
[-P resource-file] [ pattern...]
```

where:

-h prints usage information.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s lists all the files sorted by tape ID and position on the tape.

-t lists all the tapes registered with the HSM.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Examples

Example 1 This example lists all the files registered with the HSM:

```
% mdshsmdir
TapeID     Len(bytes) FNum  BlockNum FileName
------     ---------- ----  -------- --------
000001            256    1        1 /mds/movies/oracle1.mpg
000002            256    1        1 /mds/movies/oracle2.mpg
000003            256    1        1 /mds/movies/oracle3.mpg
000001            256    2        3 /mds/movies/oracle4.mpg
000002            256    2        3 /mds/movies/oracle5.mpg
```

where:

TapeID	is the name of the tape.
Len(bytes)	is the length (in bytes) of the file.
FNum	is the file-number position a file has on the tape.
BlockNum	is the first block number on the tape to which the file was written.
FileName	is the name of the file.

Example 2 This example lists all the files matching "`oracle1*`" in the HSM:

```
% mdshsmdir "oracle1*"
TapeID     Len(bytes) FNum  BlockNum FileName
------     ---------- ----  -------- --------
000001            256    1         1 /mds/movies/oracle1.mpg
```

Example 3 This example lists all the files registered with the HSM, sorted by tape:

```
% mdshsmdir -s
TapeID     Len(bytes) FNum  BlockNum FileName
------     ---------- ----  -------- --------
000001            256    1         1 /mds/movies/oracle1.mpg
000001            256    2         3 /mds/movies/oracle4.mpg
000002            256    1         1 /mds/movies/oracle2.mpg
000002            256    2         3 /mds/movies/oracle5.mpg
000003            256    1         1 /mds/movies/oracle3.mpg
```

Example 4 This example lists all the tapes registered in the HSM:

```
% mdshsmdir -t
Volume Mount  BlockSz TotalBlks  UsedBlks  RsrvBlks FNum Plays  TapeID  Fmt
------ -----  ------- ---------  --------  -------- ---- ------ ------  ---
movies UNMNT     8192        10         0         4    0      0  000001 TAR
movies UNMNT     8192        10         0         4    0      0  000002 TAR
movies UNMNT     8192        10         0         2    0      0  000003 TAR
```

where:

Volume	is the name of the volume associated with the tape.

Mount	is the mount status of the tape. **MNT** indicates that the tape is mounted in a tape device. **UNMNT** indicates that the tape is not loaded in a tape device and is in the tertiary storage device.
BlockSz	is the block size (in bytes) defined for the tape.
TotalBlks	is the total number of blocks available on the tape.
UsedBlks	is the number of blocks that are currently being used to store data.
RsrvBlks	is the number of blocks that have been reserved for backup and copying files from the MDS volume.
Plays	is the number of times the tape has been restored or backed up.
Fmt	is the format of the files stored on the tape.

Related Commands

mdshsmctl

mdslock

mdslock locks an MDS file in read-only mode, where it can be read but not written.

Syntax

```
mdslock [-h] [-V] [-T] [-R resource-statement] [-P resource-file]
mds-filename...
```

where:

-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
mds-filename	specifies one or more MDS files to be locked.

Usage Notes

mdslock provides protection from inadvertently overwriting files. For example, you can lock a content file so that another user does not overwrite it while you are playing it. In a production environment you may want to lock all the content and tag files in the MDS.

To determine whether a file is locked in read-only mode, use **mdsdir**.

Once a file is locked in read-only mode, you must unlock it with **mdsunlock** so that the file can be both read and written. Restarting **mdsdirsrv** does not unlock a file.

Example

This example locks the file **oracle1.mpg** in read-only mode:

```
% mdslock /mds/video/oracle1.mpg
```

Related Commands

mdsunlock

mdsrebuild

mdsrebuild rebuilds data onto a new disk after a disk failure or onto a disk used as a spare. You can rebuild data only if there are no other failures in the RAID set containing the disk.

Syntax

```
mdsrebuild -f voltab [-h] [-V] [-T] [-R resource-statement]
[-P resource-file] disk
```

where:

-f identifies the **voltab** file which defines the volume containing the disk, such as $ORACLE_HOME/vs30/admin/voltab. You must specify **-f**.

-h prints usage information.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

disk is the disk to be rebuilt.

Usage Notes

All volumes listed in the **voltab** file that contain the specified disk will be rebuilt by **mdsrebuild**.

The MDS can support only one rebuild operation at a time. If you have more than one failed disk in the same MDS volume set, you cannot rebuild them with concurrent **mdsrebuild** commands; you must wait for one command to return before issuing the next.

Please note, that if you run **mdsrebuild** with other MDS utilities, the **maxbw** parameter in the **voltab** file must be set or the disk I/O of the other MDS utilities may fail.

When a disk needs to be rebuilt, **mdsdirsrv** will display the message “rebuild disk *disk_name* immediately.” While **mdsdirsrv** will place the disk in rebuild mode and mount the volume, you should rebuild the disk as soon as possible.

Example

Example This example rebuilds the disk **/dev/rdsk/c2t4d0t0** using the MDS volume specification from the specified **voltab** file:

```
% mdsrebuild -f $ORACLE_HOME/vs30/admin/voltab /dev/rdsk/c2t4d0t0
```

Related Commands

mdsdiskmode

mdsrename

mdsrename renames an existing file in the MDS.

Syntax

```
mdsrename [-h] [-V] [-T] [-R resource-statement] [-P resource-file]
mds-filename1 mds-filename2
```

where:

-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
mds-filename1	is the source file to be renamed.
mds-filename2	is the destination file or the new name. If a file by this name already exists, mdsrename overwrites it. You cannot use mdsrename to move a file from one MDS volume to another.

Example

This example renames the file **oracle1.mpg** in the volume video to **first.mpg**:

```
% mdsrename /mds/video/oracle1.mpg /mds/video/first.mpg
```

Related Commands

mdscopy, **mdsdelete**

mdstar

mdstar loads files from the MDS to a tar archive or from a tar archive to the MDS, or lists files contained in a tar archive.

Syntax

```
mdstar {-c|-x|-t} [-h] [-T] [-V] -f device-name [-b integer]
[-p volume-name] [-l integer] [-R resource-statement] [-P resource-file]
files...
```

where:

-b specifies the blocking factor to be used. If you do not specify a blocking factor **tar** defaults to 20.

-c creates a tar archive and writes the named files to the archive.

-f the device-name to use as an archive.

-h prints usage information.

-l length of a file segment.

-p the volume to place files into during extraction.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-t lists the files contained in a tar archive.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

-x extracts a file from a tar archive to the MDS volume specified.

Usage Notes

The **mdstar** utility can be used without enabling the Hierarchical Storage Management (HSM) service.

When extracting from a tar archive, you must use the same blocking factor that was specified when the archive was created. For this reason you must specify the correct blocking factor for your server platform. Refer to your server platform's documentation for the correct value. The default blocking factor for **mdstar** is 20.

If **mdstar** is terminated abnormally, it marks the new files as invalid. An invalid file will not play properly. You can identify invalid files with the **mdsdir** utility. You can then remove them with the **mdsdelete** utility and reload them with **mdstar**.

Segmenting Large Files

To load a file larger than the supported capacity of the host file system, specify a limit size value with **-l**.

If you load a file larger than the limit size specified with **-l**, **mdstar** divides the file into multiple file segments, each no larger than the limit size, and loads each separately. Names of segment files have this form:

__integer1_integer2_file
where:

integer1 is a hexadecimal integer representing the byte position in the original file where this file begins.

integer2 is hexadecimal integer representing the size of the original file in bytes.

filename is the name of the original file.

For example, **mdstar** divides a 5 GB file named **oracle.mpg2** into these 3 files when the tape archive is created:

```
__0_140000000_oracle.mpg2
__7fffffff_140000000_oracle.mpg2
__ffffffff_140000000_oracle.mpg2
```

If your host file system does not support files of 2 GB or greater, you can create files using *__integer1_integer2_file.ext* naming format, put them on tape, and **mdstar** will concatenate them into one large file named *file.ext* when you load them into a MDS volume.

Examples

Example 1 This example archives the file **/mds/volume1/oracle.mpg** to the device **/dev/rmt/0**.

```
% mdstar -c -T -f /dev/rmt/0 /mds/volume1/oracle.mpg
a oracle.mpg, 1610612789 bytes, 1 segment(s), 293 kb/s
```

This example uses the default blocking factor of 20. To ensure that you are using the correct blocking factor, refer to your server platform documentation.

Note that this example and others in this section use the **-T** option. This option enables verbose mode, which lists additional information about actions taken on files.

Example 2 This example uses a wildcard search to extract all files with the **.mpg** extension and places the files in the MDS volume **video**:

```
% mdstar -x -b 128 -f /dev/rmt/0 -p /mds/video "*.mpg"
```

This archive was originally created with a blocking factor of 128. You must always extract or list the contents of an archive with the same blocking factor that was used to create it.

Example 3 This example lists the contents of the archive device **/dev/scsi/rmt7**:

```
% mdstar -t -f /dev/rmt/0
oracle1.mpg, 1610612789 bytes, 3145729 blocks, 1 segments, Feb 12
11:18:26
oracle2.mpg, 3610612 bytes, 7052 blocks, 1 segments, Feb 12 11:19:05
```

When listing the contents of an archive you must use the blocking factor the archive was created with.

Example 4 This example archives all files in **/mds/volume1** with the extension **.mpg** to a file:

```
% mdstar -c -T -f /tmp/film.tar "/mds/volume1/*.mpg"
a star_wars.mpg, 1610612789 bytes, 1 segment(s), 278 kb/s
a clip1.mpg, 3610612 bytes, 1 segment(s), 289 kb/s
```

Example 5 This example extracts files from the archive **film.tar** and places them in the MDS volume **volume1**.

```
% mdstar -x -f /tmp/film.tar -p /mds/volume1
```

For more information refer to your UNIX tar documentation.

Related Commands

mdscopy, **mdsdir**, **mdsdelete**

mdsundelete

mdsundelete restores an MDS file that has been previously removed with the **mdsdelete** utility.

Syntax

```
mdsundelete [-h] [-V] [-T] [-R resource-statement]
[-P resource-file] mds-filename...
```

where:

-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
mds-filename	specifies one or more MDS files to be restored.

Usage Notes

When **mdsdelete** removes a file, it does not overwrite the file unless the **-z** option is specified; it simply makes the file invisible to OVS server processes. If you inadvertently remove a file from the MDS volume, you can restore the file with mdsundelete unless the file has been overwritten by another file that has since been loaded into the same volume. You can identify removed files that have not yet been overwritten with **mdsdir**. The overwriting of invisible removed files by newly loaded files is not predictable.

Example

This example removes two files with **mdsdelete** and restores one of them with **mdsundelete**:

```
% mdsdelete   /mds/video/oracle1.mpg /mds/video/oracle2.mpg
% mdsundelete /mds/video/oracle1.mpg
```

Related Commands

mdsdelete

mdsunlock

mdsunlock unlocks a file in the MDS that has been locked with the **mdslock** utility**.**

Syntax

```
mdsunlock [-h] [-V] [-T] [-R resource-statement] [-P resource-file]
mds-filename...
```

where:

-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
mds-filename*e*	specifies one or more MDS files to be unlocked.

Usage Notes

You can identify MDS files locked in read-only mode with the **mdsdir** utility.

When a file is unlocked, it can be written to the corresponding volume if the volume has been mounted in read-write mode. An unlocked file cannot be written to by a read-only **mdsdirsrv**.

Example

This example unlocks the file **oracle1.mpg** located in volume **video**:

```
% mdsunlock /mds/video/oracle1.mpg
```

The file can now be both read and written.

Related Commands

mdslock

mdsvolinit

mdsvolinit writes the definition of an MDS volume, as it appears in a **voltab** file, to each of the volume's disks. It can be used to either initialize a volume specification and / or to clear a volume's table of contents.

Syntax

```
mdsvolinit [-b disk...|-s] [-t] [-h] [-V] [-T] [-R resource-statement]
[-P resource-file] -f voltab volume...
```

where:

-b specifies broken or failed disks.

-f identifies the **voltab** file containing the volume definition. You must specify the **voltab** file.

-h prints usage information.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s initializes volume specification. This should be used whenever a new volume is created.

-t indicates that the table of contents is to be initialized.

CAUTION: The -t option removes all content from the specified volume. **This option should only be used when you create a new volume or you want to clear all files from a volume.**

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

voltab specifies the voltab file to be used, such as $ORACLE_HOME / vs30 / admin / voltab.

volume specifies the volume(s).

Usage Notes

When **mdsdirsrv** starts, it returns an error message if the volume definitions of the **voltab** file do not match those of the disk. You can then examine the message to determine why they do not match:

- If you have intentionally modified the volume specification in the **voltab** file since last starting **mdsdirsrv**, use **mdsvolinit** to write the new volume definitions to disk. Examples of this include changing any parameters such as the name of the volume or the addition of the spare disk.

  Always use the **-s** and **-t** switches together. Running **mdsvolinit** will initialize the disk according to the new **voltab** and erase the current volume.

- If the volume definition in **voltab** has been inadvertently modified or otherwise corrupted since **mdsdirsrv** was last started, modify **voltab** to correct the volume definition.

- Under limited circumstances, you may use **-s** without **-t** for one of the following purposes:

  – to change the name of a volume

  – to add, delete, or change the name of the spare disk(s)

  – to migrate a volume from one version of the video server to another

  ---

  Warning: For all other **voltab** changes, such as changing the raid size, you must reinitialize your volume with both the **-s** and **-t** options. This will destroy all the files in your current volume.

  You will be prompted for a yes/no reply when using the **-s** flag.

- If a new volume is created, or you want to clear all files from a volume, use **-s** and **-t** to initialize the volume. You will be prompted for a yes/no reply during each phase of the initialization.

After resolving the differences between **voltab** and the disks, restart mdsdirsrv.

If you try to write volume definitions to a failed disk, **mdsvolinit** returns an error. If a volume contains a failed disk and you want to write the volume definition to its other disks, specify the failed disk with the **-b** switch. Later if you replace the failed disk and rebuild data onto the replacement disk with the **mdsrebuild** utility, **mdsrebuild** also writes the volume definition to the disk, so you need not use **mdsvolinit**.

Example

This example initializes the volumes video1 and video2 using the **voltab** file located in the directory $ORACLE_HOME/vs30/admin:

```
% mdsvolinit -s -t -f $ORACLE_HOME/vs30/admin/voltab video1 video2
```

> **Note:** You must initialize each MDS volume with this syntax at the time of creation. If a volume is not initialized, **mdsdirsrv** will not recognize it a OVS start-up. Be aware that initializing an MDS volume erases all data from the volume.

Related Commands

mdsrebuild

mdsvolstat

mdsvolstat empirically measures an MDS volume for the following information and statistics:

- general MDS volume statistics
- the value for **maxbw**
- sample disk I/O rates

> **Note: mdsvolstat is not supported on MPP systems.**

Syntax

```
mdsvolstat [-s] [-d] [-a] [-h] [-V] [-T] [-f voltab] [-c integer]
[-m integer] [-b integer] [-i disk-name] [-r integer]
[-R resource-statement] [-P resource-file] [volume-name...]
```

where:

-a obtains the **maxbw** value for an MDS volume. For information on **maxbw** see Chapter 11, "Oracle Video Server Components."

-b specifies the video bit rate to use with **-a,** in bps (refer to the **vspump** section of Chapter 11, "Oracle Video Server Components" of this guide).

-c use this option with **-a** to determine the length in seconds of the disk I/O simulation; the default is 15 seconds.

-d produces an average disk I/O time in microseconds for a specified MDS volume.

-f identifies the **voltab** file containing the volume definition.

-h prints usage information.

-i averages the specified disk's I/O time in milliseconds. Use this option in conjunction with **-d**.

-m determines the maximum number of I/Os when used with **-d**; the default is 15.

-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-r	determines how often messages are logged, in milliseconds, when using **-a** to obtain the **maxbw** value. This option provides additional information about how a system performs under a given load.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-s	creates a list of statistics about the specified volume.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes. Use the **-T** option to see additional logging.
-V	prints a version banner.
volume-name	an MDS volume about which to report. You may specify multiple volume names, or no volume name to operate on all volumes.

Usage Notes

mdsvolstat performs intensive I/O operations when measuring an MDS volume. For this reason **mdsvolstat** should always be used on its own, and never on a volume being accessed by other OVS utilities or clients (such as the video pump).

When testing a volume for its **maxbw** value, **mdsvolstat** can be used with **-b** to test a volume with a specific video bit rate. This allows you to compare the number of concurrent video streams that can be provided at different bit rates by changing the bit rate value of the video pump (**vspump**) and encoding your content at different rates.

When sampling the average I/O time of a disk with the **-d** option, you can set the number of disk I/Os with **-m**. If **mdsvolstat** overloads a disk and returns an error, reduce the number of disk I/Os. Specifying more I/Os produces a more accurate average of I/O times.

mdsdirsrv should not be running when using **mdsvolstat**. To use **mdsvolstat**, start only Oracle Media Net. This allows **mdsvolstat** to "write" to the MDS.

For information on starting only Oracle Media Net refer to Chapter 4, "Operating the Oracle Video Server" of the *Oracle Video Server Administrator's Guide*.

Examples

Example 1 The **maxbw** value determines the maximum available bandwidth that an MDS volume can deliver. An accurate **maxbw** value is crucial when determining the number of concurrent video streams that the server can deliver.

This example obtains the **maxbw** value for the volume **movies**:

```
% mdsvolstat -a -f $ORACLE_HOME/vs30/admin/voltab movies
Disk average = 15643 usecs, theoretical maxbw is 160.88 mbs.

Attempting a maxbw of 10.05 bytes/usec, 80 mbs
   52 streams, 208 IO buffers, 13631488 bytes RAM
Attempting a maxbw of 10.05 bytes/usec, 80 mbs
   52 streams, 208 IO buffers, 13631488 bytes RAM
Attempting a maxbw of 5.03 bytes/usec, 40 mbs
   26 streams, 104 IO buffers, 6815744 bytes RAM
Attempting a maxbw of 5.03 bytes/usec, 40 mbs
   26 streams, 104 IO buffers, 6815744 bytes RAM

..... etc.
Attempting a maxbw of 9.43 bytes/usec, 75 mbs
   49 streams, 196 IO buffers, 12845056 bytes RAM
Attempting a maxbw of 9.11 bytes/usec, 72 mbs
   47 streams, 188 IO buffers, 12320768 bytes RAM
Attempting a maxbw of 9.27 bytes/usec, 74 mbs
   48 streams, 192 IO buffers, 12582912 bytes RAM

Recommended maxbw setting for volume movies is: 55 mbs
```

mdsvolstat will attempt various I/O rates until it determines the correct **maxbw** value through a process of elimination.

Example 2 You can use **mdsvolstat** to list general statistics about a volume with the **-s** option. This example displays statistics for the volume **video**.

```
% mdsvolstat -s -f $ORACLE_HOME/vs30/admin/voltab video
------------------------------------------------------------------------
Volume name:                            video
Parity:                                 yes
IO size, individual disk:               65536 b, 64 k
IO size, full stripe:                   327680 b, 320 k
IO size, full stripe minus parity:      262144 b, 256 k
Number of normal disks:                 5
Number of spare disks:                  1
```

```
Disks per raidset:                      5
Number of raidsets in volume:           1
TOC size:                               262144 b, 256 k
Maximum files possible in volume:       4095
Raw capacity of each disk:              1048576 b, 1024 k, 1 m
Available capacity of each disk:        786432 b, 768 k
Stripes per raidset/disk:               12
Total stripes in volume:                12
Total volume capacity:                  2883584 b, 2816 k
```

Example 3 This example measures the I/O speed of a random disk in the volume **video**. The time of the I/O operation is recorded in microseconds. **mdsvolstat** first sends I/O data of the size specified individually and averages the time, and then in batches. Batch numbers should be the same or lower than the individual I/O times. If they are not, the device drivers used by the disks are not as efficient as they could be.

```
% mdsvolstat -d -f $ORACLE_HOME/vs30/admin/voltab video
Disk Name                        IO size    Num IOs        Avg (us)
-------------------------------------------------------------------------
/dev/rdsk/c1t2d0s3                512         15           10066
/dev/rdsk/c1t2d0s3                512         15            6517(batch)
/dev/rdsk/c1t2d0s3               4096         15            9301
/dev/rdsk/c1t2d0s3               4096         15            7331(batch)
/dev/rdsk/c1t2d0s3              65536         15           21935
/dev/rdsk/c1t2d0s3             65536         15          15126(batch)
```

Related Commands

mdsdir

13

Stream Service Utilities

The stream service utilities perform special operations on content files in the Oracle Media Data Store (MDS) or a host file system. These utilities include:

vscontdel	Deletes logical content titles and associated database objects from the database storing logical content information.
vscontreg	Re-registers tag files with the database. If you move content, including tag files, from one MDS volume to another, you can reregister the tag files with the existing logical content database objects.
vsdbbuild	Creates initial user and schema in database.
vsgentag	Registers an MPEG or raw key frame file. Unlike **vstag**, **vsgentag** creates a *null tag file*, which does not allow rate control operations. Use **vsgentag** to register content where rate control is not essential or proprietary user codecs are used.
vsmkosf	Converts AVI and WAV files into Oracle Streaming Format (OSF) files, which are optimized for streaming. This utility also registers the OSF file with the MDS and database.
vsmpegchk	Verifies the integrity of an MPEG-2 transport file.
vstag	Registers MPEG and OSF files with the Oracle MDS and database.
vstagpatch	Edits tag files for registered content files.
vstagprint	Prints the tag file information associated with registered content files.

vscontdel

vscontdel removes logical content titles and associated database objects from the database storing logical content information. If you delete content files and tag files from the MDS, you must also delete the logical content titles associated with those files. **vscontdel** allows you to do this using the OVS command line interface. VSM automatically does this for you when deleting content through VSM.

Syntax

`vscontdel [-h] [-P] [-R] [-T] [-V] [-c] [-t] logical-content-title`
where:

-c	deletes the tag file and the content file
-h	prints usage information.
-t	specifies that **vscontdel** also delete the tagfile in the MDS associated with the logical content title when possible.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
logical-content-title	specifies the file to delete.

Usage Notes

The **vscontdel** utility removes logical content titles and associated database objects from the database storing logical content information. You can specify that **vscontdel** delete tag files associated to the logical content title with the **-t** option.

Note that **vscontdel** only deletes clips, physical content descriptions, and tag files if they are associated with only the logical content title being deleted. If these database objects are used by other logical content titles **vscontdel** does not remove them. If you try to delete database objects in use by other logical content titles **vscontdel** returns an error message stating that it cannot delete the specified objects.

Examples

Example 1 This example deletes the logical content title "*Best Cartoon's of 1997*" from the OVS database storing logical content information. When deleting a logical content title, **vscontdel** examines the clips and physical content descriptions associated with the title being deleted. If the objects are associated to other logical content titles **vscontdel** will not remove them:

```
vscontdel "Best Cartoon's of 1997"
```

Example 2 This example uses **-t** to specify that **vscontdel** remove all database objects associated with the logical content title **Video Training** as well as the tag files stored in the MDS. If any of the clips or physical content descriptions are used by another logical content title, **vscontdel** will return an error message stating that it cannot delete the associated database objects.

```
vscontdel -t "Video Training"
```

Related Commands

vscontsrv, **vscontreg**

For Further Information

You will find information on:

- the logical content model and the creation of logical content titles in the document *Getting Started with Oracle Video Server Manager.*
- implementing the logical content service in Chapter 9, "Configuring the Logical Content and Scheduling Services".

vscontreg

vscontreg allows you to update logical content entries in the database when moving content from one OVS system to another.

Use **vscontreg** when:

- adding a logical content database to an existing OVS system
- moving your OVS installation from one operating system type to another (i.e. Sun Solaris 2.5.1 to HP-UX 10.01)

Syntax

`vscontreg [-h] [-P] [-R] [-T] [-V] [-d] tagfile`
where:

-d specifies that **vscontreg** delete existing database objects using the same logical content title

-h prints usage information.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Usage Notes

The **vscontreg** utility allows you to update logical content entries when moving content from one Oracle Video Server system to another. If, for example, you migrate your Oracle Video Server system to a more powerful server with an expanded MDS file system, you can update your existing logical content database by re-registering your tag files with **vscontreg**.

vscontreg only creates simple logical content entries, similar to those produced by **vstag**. It will not retain any complex logical content from another OVS instance.

Examples

Example 1 This example reregisters the file **oracle1.mpi** stored in the MDS volume **volume2**:

```
vscontreg /mds/volume2/oracle1.mpi
```

Example 2 This example reregisters all tag files ending with the extension **.mpi** in the MDS volume **video**:

```
vscontreg "/mds/video/*.mpi"
```

Example 3 In this example, **vscontreg** reregisters the file **oracle1.mpi** while deleting any database objects using the same name that the tag file uses:

```
vscontreg -d /mds/video/oracle1.mpi
```

Related Commands

mdscopy, **mdsdelete**, **vscontsrv**, **vscontdel**, **vstagpatch**

For Further Information

You will find information on:

- the logical content model and the creation of logical content titles in the document *Getting Started with Oracle Video Server Manager.*
- implementing the logical content service in Chapter 9, "Configuring the Logical Content and Scheduling Services".

vsdbbuild

The **vsdbbuild** utility creates a user account and schema for Oracle Video Server's database-dependent features. The database-dependent features are:

- Logical Content Service
- Scheduling Service

These services are described in Chapter 9, "Configuring the Logical Content and Scheduling Services".

Syntax

```
vsdbbuild [-nDhVTU] [-s systempwd] [-u usertblspc] [-t temptblspc] connect_string
```

where:

-D	Removes the user account prior to recreating the schema. If the user account exits in the database **vsdbbuild** prompts for confirmation prior to dropping the account. Note that the user will only be removed if the oracle system password is specified with **-s**.
-h	Prints help.
-n	Prevents **vsdbbuild** from executing the specified commands. Use this option in conjunction with **-T** to list SQL *Plus commands without modifying the user account and schema in the database.
-s *system password*	Creates a new user account with the information supplied in the connect string to assign a username and password. You must supply the Oracle system password as an argument.
-T	Echoes the SQL *Plus command used to execute **vsdbbuild**'s actions.
-t *tablespace name*	Uses the specified temporary tablespace name for the new user account. The default is TEMP. The **-t** option is only valid when creating a new user account with the **-s** option.
-U	Updates the user account and schema. Performs schema upgrade and refreshes PL/SQL only. Does not create tables.

-u *tablespace name*	Specifies the tablespace to use when creating a new user account. When not specified **vsdbbuild** uses the default tablespace name USERS.
-V	Prints version information.
connect string	The connect string identifies the user account to create or modify and the database to use for that account. The syntax of the connect string is ***username/password@alias***: ***username*** The user name the account is created for. ***password*** The password to use with the account. ***alias*** The TNS alias identifying the database as defined in the **tnsnames.ora** file.

Usage

The **vsdbbuild** utility allows you to:

- create both a new database user account and tables for Oracle Video Server.
- create a database schema for Oracle Video Server under an existing user account
- specify a tablespace name to use with a new user account
- specify a temporary tablespace name to use with a new user account

The **vsdbbuild** utility creates the user account and schema for all database dependent services. **vsdbbuild** accepts the following input as arguments to create the user account and schema:

- Oracle system password
- name to use for the Oracle Video Server tablespace
- name to use for the Oracle Video Server temporary tablespace
- username for the database account
- password for the database account
- database connect string

Before using **vsdbbuild** to create a new user account and schema, ensure that you have the correct information to create the database account under which you will operate Oracle Video Server.

Creating New Accounts

Before creating a database user account, ensure that you have adequate system privileges on the instance of the Oracle database in use by Oracle Video Server.

To create a new user account, specify the **-s** option with the Oracle system password. The account you create uses the username and password you specify in the connect string. By default, **vsdbbuild** uses the name USERS for the tablespace. You can specify a different tablespace name with the **-u** option.

The following example creates the user account **ovs** on the database instance **ovsdb**:

```
% vsdbbuild -s oracle ovs/ovs@ovsdb
```

Since this example omits the **-u** option, **vsdbbuild** assigns the default tablespace name USERS.

To Use vsdbbuild

The following environment must exist to create a database account with **vsdbbuild**:

- The server running Oracle Video Server and the instance of the database must have a TCP / IP connection
- The database must have XA transaction processing enabled
- The database must be on-line with a listener that is accepting connections
- Oracle Video Server must have a SQL *Net configuration file.

Examples

Example 13–1 Creating a New User Account and Schema

```
% vsdbbuild -s oracle -u users -t temp ovs/ovs@ovsdb
```

This example creates a user account and schema with the following attributes:

system password	**oracle**
tablespace name	**USERS**
temporary tablespace name	**TEMP**

user name	**ovs**
password	**ovs**
TNS alias of the database	**ovsdb**

Note that you can only specify **-u *tablespace*** and **-t *temporary tablespace*** when creating a new user account with **-s *system password***.

Example 13–2 Creating a Schema for an Existing User Account

```
% vsdbbuild ovs/ovs@ovsdb
```

This example creates the Oracle Video Server database schema for the user with the connect string **ovs/ovs@ovsdb**.

Example 13–3 Removing a User Account

```
% vsdbbuild -s oracle -D ovs/ovs@ovsdb
```

To remove (or drop) a user account, specify the **-D** option with the user's connect string. Note that you must also provide the Oracle system password to remove a user account.

vsgentag

Use the vsgentag utility to register content with a null tag file. Null tag files do not allow rate control operations to be performed during playback. Use **vsgentag** to register:

- raw key frame files (such as Motion-JPEG) which are not supported by the **vstag** utility.
- files where rate control is not desired.

> **Note:** Generating a null tag file for a raw key frame file enables OVS to stream an unsupported format to a client. However, the client must have the appropriate decoder to play the file.

Syntax

```
vsgentag [-h] [-V] [-T] [-d description] [-f compression-format]
[-t transport-wrap] [-R resource-statement] [-P resource-file]
bit_rate input-filename output-filename
```

where:

-d writes a descriptive comment into the tag file. You can read this comment later with **vstagprint** or change it with **vstagpatch**. If you are registering a group of files, **vstag** writes this comment into every tag file.

-f specifies the compression format being registered. The formats are:

key specifies raw key frame. This encompasses OSF and any stateless contiguous frame compression format (such as Motion JPEG).

mpg1 specifies an MPEG–1 system stream.

m2t specifies MPEG–2 transport (default).

Note: If you specify a format not listed here, you must make sure that the client associates the format identifier with the proper codec and has the decoder necessary to play the file

-h prints usage information.

-P	specifies the resource file. See “Using Resource Descriptors” on page 11-3 of this guide.
-R	sets the specified resource descriptor. See “Using Resource Descriptors” on page 11-3 of this guide.
-t	specifies the transport type of the input file:
gen	generic framing. Do not use this option in this release.
m2t	MPEG-2 framing, used for some set-top boxes (default)
raw	produces an unframed output file. This should be specified when creating tag files for OSF and MPEG–1 (system stream) content. **Note**: If you specify a format not listed here, you must make sure that the client associates the format identifier with the proper codec and has the decoder necessary to play the file.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
bit_rate	the bit rate of the content file in bit per second (bps). For example 1536000 bps.
input-filename	specifies the content file(s) to be tagged.
output-filename	specifies the output tag file name and location.

Usage Notes

vsgentag does not examine the content when it creates the tag file. Instead, it assumes that the information you provide is accurate and generates the tag file based on this information. For this reason, you must provide **vsgentag** with accurate information or the content file might not play properly.

Examples

Example 1 This example registers the OSF file **video1.avi** and creates the tag file **video1.mpi**, specifying a compression format of type **key** with **-f**, a transport of type **raw** with **-t**, and a bit rate of 1,200,000 bps:

```
vsgentag -f key -t raw 1200000 /mds/video/video1.osf /mds/video/video1.mpi
```

Example 2 In this example, the tag file **oracle1.mpi** is generated for the MPEG–1 file **oracle1.mpg** encoded at 1.536 Mbps, specifying a compression format of type **mpg1** with **-f**, a transport of type **raw** with **-t**, and a a bit rate of 1,536,000 bps:

```
vsgentag -f mpg1 -t raw 1536000 /mds/video/oracle1.mpg \
/mds/video/oracle1.mpi
```

Related Commands

vsmkosf, **vstag**, **vstagpatch**, **vstagprint**

vsmkosf

The **vsmkosf** utility converts AVI and WAV files into OSF files and registers the new OSF file with the Oracle MDS and the database. AVI and WAV files are not optimized for streaming and therefore must be converted to OSF files when loaded into OVS.

Syntax

`vsmkosf [-hshVT] [-f flags][-i init-record-rate] [-n] [-s] [-t record-time] [-R resource-statement] [-P resource-file] input-filename output-filename`
where:

-f tells **vsmkosf** the types of frames to find and tag.

- **i** tags MPEG I-frames. You should use this flag whenever you generate a tag file. This is the default parameter.
- **a** tags an audio-only file in such a way to make the audio file seekable

-h prints usage information.

-i specifies the frequency at which init records should be inserted into the OSF content file. For example, if **-i** is set to 5, an init record will be inserted for every 5 data records. If **-i** is set to 0, the default, then one init record is inserted at the beginning of the OSF file.

-n disables content registration. Use the **-n** flag when you want to convert a file to OSF, but you do not want the created OSF file to be registered with the MDS or database. When the **-n** flag is used, no tag file or database objects are created.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s prints out the OSF file values that would be used if you converted the specified AVI file into an OSF file. *This command does not create an OSF or tag file.*

-t specifies the time interval (from 50 to 2000 milliseconds) of each OSF data record. If **-t** is not specified, the utility will find the optimal value.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

Usage Notes

You do not need to provide extensions for your output files; **vsmkosf** appends the appropriate extension to each output file. The **vsmkosf** utility creates two output files:

- .osf — streamable OSF content file
- .mpi — a tag file which calls the streamable .osf file

Init and Data Records

OSF files are composed of fixed-time blocks, called *records*. There are two types of records, *data* and *init*. Data records include audio, video, and header information. Init records contain setup information, including which audio and video codecs the client needs to decode the file. OSF files must have at least one init record.

Use the **-t** flag to specify the size (in milliseconds) of data records.

Use the **-i** flag to specify the frequency at which init records should be inserted into the OSF content file.

Examples

Example 1 This example converts and registers the file **oracle1.avi** in the local file system and copies it to the MDS volume **video**:

```
vsmkosf ./oracle1.avi /mds/video/oracle1
```

This example creates the files **oracle1.mpi** and **oracle1.osf**.

Example 2 This example uses wildcards to convert and register multiple AVI files:

```
% vsmkosf ./*.avi /mds/video
```

Since this example converts multiple files, the last argument is the output MDS volume, not the file name.

Example 3 This example converts and registers the file **oracle1.avi,** creating an OSF file that has a data record length of 1000 milliseconds (**-t**) and inserts 1 init record for every 5 data records (**-i**):

```
vsmkosf -t 1000 -i 5 ./oracle1.avi /mds/video/oracle1
```

Example 4 This example, which uses the **-s** flag, prints out the OSF file values that would be used if you converted the file **oracle1.avi** to an OSF file. *This command does not create an OSF or tag file.*

```
vsmkosf -s /mds/video/oracle1.avi
```

Output might look like:

```
video type is UCOD
audio type is WV071C
video frequency:      15 / 1
audio frequency:     441 / 20
record frequency:      5 / 1
frames per record:     3 / 1
samples per record: 4410 / 1
min audio vlock size:  1
init records frequency:0
total number of records: 7890
```

where

video type is the video codec that was used to create the AVI file. Sample values are:

- **UCOD —** Iterated Systems ClearVideo
- **IV41** — Intel Indeo

audio type is the audio codec that was used to create the AVI file. Sample values are:

- **WV071C**— Lucent Technologies SX-7300
- **WV6200 —** Voxware MetaSound
- **WV0100** — PCM (16-bit mono, uncompressed)

video frequency is the video frame rate (frames per second) that was used to create the AVI file.

audio frequency is the audio sampling rate (in KHz) that was used to create the AVI file.

record frequency is the time interval for each OSF record. For example, a value of **5 / 1**, or 5 records per second, would indicate that there will be one record every 200 milliseconds (equivalent to the **-t** flag value).

frames per record is the number of video frames in each record.

samples per record is the number of audio samples in each record.

min audio block size is the minimum audio sample size (in bytes)

init records frequency is the rate at which init records are inserted into the OSF file (equivalent to the **-i** flag value).

total number of records is the total number of records that will be created for this OSF file.

Example 5 This example converts the file **oracle1.avi** into an OSF file, but *does not* register the OSF file with the Oracle MDS or database, therefore not creating an .mpi file. **vstag** can be used later to generate the .mpi file for this .osf file:

```
% vsmkosf -n ./oracle1.avi /mds/video/oracle1.osf
```

This example creates the file **oracle1.osf**.

Related Commands

vscontdel, vscontreg, vsgentag, **vstag**, **vstagpatch**, **vstagprint**

vsmpegchk

vsmpegchk verifies the integrity of an MPEG–2 transport file, and returns a message indicating whether the file is valid or corrupted.

Syntax

vsmpegchk [-h] [-P] [-R] [-s] [-T] [-V] *filename...*
where:

-h prints usage information.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s shows the type and quantity of each packet type in the file.

-T enables verbose mode. Verbose mode returns information about actions performed by processes.

-V prints a version banner.

filename is a file to be verified. You may specify multiple files.

Usage Notes

vsmpegchk can be used only with MPEG-2 transport. Verify that you are using content wrapped in MPEG-2 transport prior to using **vsmpegchk**.

NOTE: A file may be deemed playable by **vsmpegchk**, but not by **vstag**. When **vstag** generates a tag file, it examines the content file to a much greater degree than does **vsmpegchk**.

Examples

Example 1 This example verifies the file **oracle1.mpg2** in the MDS volume **video**:

```
% vsmpegchk /mds/video/oracle1.mpg2
oracle1.mpg2: ok
```

Example 2 This example verifies the integrity of the files **oracle1.mpg2** and **oracle2.mpg2** in the volume video:

```
 % vsmpegchk /mds/movies/oracle1.mpg2 /mds/movies/oracle2.mpg2
oracle1.mpg2: ok
oracle2.mpg2: error at position 188: 23 45 31
```

The returned message shows that:

- oracle1.mpg2 is valid.
- oracle2.mpg2 has invalid MPEG-2 data at address 188 and the data there is 23, 45, and 31.

Related Commands

vsgentag, **vstag**

vstag

Use **vstag** to register MPEG or OSF files.

Before you can play content files from the Oracle Video Server (OVS), you must register the file with the MDS and database (if available). For each content file that you register, you generate:

- a tag file which is stored in the same MDS as the media content file.

  Tag files contain important metadata information, such as encoding rate, duration, and data used for rate control operations (seeking, scanning, pausing) that the client application uses when accessing the video content file. Tag files must be stored in the same MDS volume as the content file.

- logical content, clip, and physical content objects that are stored persistently in the database.

> **Note:** AVI and WAV files are not optimized for streaming and therefore you must convert them to OSF files, using the **vsmkosf** utility.

Syntax

```
vstag [-h] [-i] [-V] [-T] [-e end-address] [-f flags{i}] [-s start-address
integer]
[-d description] [-D rate-control-prohibitions {p | s | k | K | n | N | f | F}]
[-E output-file-extension] [-p parallel-node-spec integer ]
[-n slave specification] [-o output_content] [-r presentation rate] [-R] [-P]
input-filename [output-filename]
```

The *filenames* argument is interpreted in one of these ways:

```
vstag [options] input-filename output-filename
vstag -E extension [options] input-filename1 [input-filename2 ...]
```

where:

-d writes a descriptive comment into the tag file. You can read this comment later with **vstagprint** or change it with **vstagpatch**. If you are tagging a group of files, **vstag** writes this comment into every tag file.

-D generates a tag file that does not allow the specified rate control operations. Rate control is the ability to play video at different speeds and in different directions, to seek in either direction, to scan in either direction, or to pause and resume playback of the video file. The rate control operations are:

- **p** pauses video.
- **s** stops video.
- **k** seeks forward (blind fast forward).
- **K** seeks backward (blind reverse).
- **n** scans forward (visual fast forward).
- **N** scans backward (visual reverse).
- **f** plays forward a frame at a time.
- **F** plays backward a frame at a time.

When specifying the above options, list the options as a string after the **-D** as shown below. This example prevents a client from moving backward in a file:

```
 % vstag -D NKF /mds/video/spleen.mpg \
/mds/video/spleen.mpi
```

Note: Do not type the backslash (\) when entering this command; it is used here as a typographical convention.

Note: Currently, OVS does not allow the prohibition of any rate control operations except for the following: pause, scan forward, and scan backward—even if these prohibitions have been specified.

-e specifies the decimal byte position of the input file at which to stop tagging. If you omit **-e**, **vstag** stops at the end of the file.

-E specifies an extension for a group of output tag files. Do not specify the period (.) as part of the extension. For example, if you used:

```
% vstag -E mpi /mds/vol1/oracle.mpg
```

the resulting tag file would be stored in **/mds/vol1/oracle.mpi.**

-f tells **vstag** the types of frames to find and tag.

i tags MPEG I-frames. You should use this flag whenever you generate a tag file. This is the default.

a tags an audio only file in such a way to make the audio file seekable.

-i specifies that **vstag** ignore any errors it may find in a file. This option allows you to tag a corrupted file and attempt to play it. Note that the file may not play correctly or at all.

-n runs parallel tagging processes in the configuration specified. This is implementation specific. Generally, this applies to MPP server systems.

-o enables you to copy a content file into the MDS and register it in a single step. Use this flag to send the file being registered to the specified location; either a directory on the host file system or to the specified MDS volume.

Example

This example uses a wildcard specification to copy and register all content files with the extension .mpg in the path `/local/video/mpeg`, and outputs both tag files and the content files to the MDS volume video:

```
% vstag -E mpi -o /mds/video /local/videos/mpeg/*.mpg
```

-p runs parallel tagging process. The number of parallel processes created is specified with the integer argument of the option. Each process tags a part of the input file.

If you omit both **-p** and **-n**, **vstag** runs in a single process.

If you use specify **-p 0**, **vstag** allocates an optimum number of processes, usually one per processor.

Example

This example runs four parallel registry processes, and registers all of the files in the volume **movies** that have the file extension **.mpg** into corresponding .**mpi** files.

```
vstag -p 4 -E mpi "/mds/movies/*.mpg
```

-r specifies the presentation rate at which the input file is encoded. Regular forward is represented by 1000, regular reverse by -1000, double speed forward by 2000, half speed forward by 500, etc. The default is 1000 and the minimums are 2 and -2. Use **-r** only if you are generating a tag file member for multi-stream rate control.

Note: Do not use the **-r** option unless directed to do so by Oracle.

-s specifies the decimal byte position of the input file at which to begin tagging. If you omit **-s**, **vstag** starts at the beginning of the file.

input-filename specifies the content file(s) to be tagged. These files must be one of the following formats:

MPEG-1 system stream
MPEG-1 or MPEG-2 elementary streams wrapped in MPEG-2 transport
OSF

output-filename specifies the output tag file name when the **-E** option is not used. (Refer to the **-E** option to see how output filenames are formed in that case.)

Note: **vstag** requires that the full MDS path of the output file be explicitly stated, or that the MDS_CWD environment variable be set to the appropriate MDS volume. For information on setting MDS_CWD refer to Chapter 7, Oracle Media Data Store Tasks and Procedures.

Examples

Example 1 This example registers the content files **oracle1.mpg** and **oracle2.mpg** in the MDS volume video as well as in the Oracle database:

```
% vstag -E mpi /mds/video/oracle1.mpg /mds/video/oracle2.mpg
```

Example 2 This example uses a wildcard search to register all of the files in the volume **video** that have the file extension **.mpg** into corresponding .**mpi** files.

```
% vstag -E mpi "/mds/video/*.mpg"
```

Example 3 This example uses the **-D** option with the **p** argument to specify that pause be prohibited when playing the file **oracle2.mpg**. Note the use of **-d**; this writes a comment to the tag file which explains that this file was registered without the ability to pause.

```
% vstag -D p -d "oracle2-no pausing" /mds/video/oracle2.mpg \
/mds/video/oracle2.mpi
```

Example 4 This example uses a wildcard specification to register all files with the extension **.mpg** in the path **/local/video/mpeg**, and outputs both tag files and the content files to the MDS volume **video**, with the **-o** option:

```
% vstag -E mpi -o /mds/video /local/video/mpeg/*.mpg
```

Related Commands

vsgentag, **vsmpegchk**, **vstagpatch**, **vstagprint**

vstagpatch

vstagpatch edits the information in a tag file.

From time to time, you might need to change the registry information for a content file. For example, you might want to move content files to another MDS volume, or change the supported rate control operations. Rather than re-register your content file, which can be a time-consuming task, you can use **vstagpatch** to edit the information in tag files.

You can use **vstagpatch** to change the following tag file properties:

- its associated content file
- descriptive comment
- bit rate
- restricted rate control operations

Note that **vstagpatch** also updates database information if used while connected to the database.

Syntax

```
vstagpatch [-c] [-s] [-p] [-v] [-h] [-V] [-T] [ -b integer ] [ -d "text" ]
[ -D { 0 | p | s | k | K | n | N | f | F} ] [ -f compression-fmt]
[ -F integer ] [-g integer] [ -l integer ] [ -L integer ] [ -m integer ]
[ -M integer ] [ -n filename ] [ -N integer ]
[ -r integer ] [ -t transport-fmt] [-w integer] input-filename
```

where:

-b specifies bit rate with which the associated content file is to be played. This value is expressed in bits per second.

Note: Changing the bit rate in the tag file does not change the encoding rate of the associated file, it simply replaces incorrect bit rate information.

-c certifies that tags are valid, but does not check against the content file itself. If **-s** is specified, **-c** is deactivated.

-d adds a descriptive comment to the tag file. The comment should be entered within single quotes. For example:

```
% vstagpatch -d 'Addendum to Presentation 3'
```

If the tag file already contains a comment, **vstagpatch** overwrites it.

-D modifies the tag file so that it does not allow the specified rate control operations:

- **0** (zero) turns off all rate control prohibitions.
- **p** pauses video.
- **s** stops video.
- **k** seeks forward (blind fast forward).
- **K** seeks backward (blind reverse).
- **n** scans forward (visual fast forward).
- **N** scans backward (visual reverse).
- **f** plays forward a frame at a time.
- **F** plays backward a frame at a time.

Note: Currently OVS does not allow the prohibition of any rate control operations except for the following: pause, scan forward, and scan backward—even if these prohibitions have been specified.

-f compression format.

-g specifies the height in pixels.

-h prints usage information

-M specifies duration in milliseconds of the associated content file.

-n associated content file. If you use **-n**, you must provide the full MDS path of the content file.

-p prints the entire tag file in the format used by **vstagprint**.

-P specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.

-R sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.

-s	if you specify **-p**, **vstagpatch** prints the header of the tag file without the entire contents.
-t	transport format.
-T	enables verbose mode, which returns additional information about action performed by processes.
-V	prints a version banner.
-w	specifies the width in pixels.
input-filename	the tag file to be modified.

Caution: The following options are currently not supported—do not use them unless so directed by Oracle.

-F	first presentation time stamp.
-l	length in bytes of associated content file.
-L	last presentation time stamp.
-m	specifies the member of the tag file to be modified if you are using multi-stream rate control.
-N	number of tags in the tag file or member.
-r	presentation rate of the associated content file. Regular forward is represented by 1000, regular reverse by -1000, double speed forward by 2000, half speed forward by 500, etc.

Usage Notes

When you play a content file, the stream service uses the tag metadata to play the content file appropriately. The metadata includes the content file's name (including its path), bit rate, and other information.

If you move or rename a content file, you must update the metadata with the new path and name using **vstagpatch** with the **-n** option.

You can use **vstagprint** to review the metadata.

Note that **vstagpatch** modifies the input file and does not accept an output file specification. If you want to modify a tag file and give it a new name, you must explicitly copy it with **mdscopy** before using **vstagpatch**.

Examples

Example 1 In this example, **oracle1.mpi** is the tag file for **oracle1.mpg**. This example uses the **mdscopy** utility to copy these files from one MDS volume to another and then uses **vstagpatch** to associate the new tag file with the new content file:

```
% mdscopy /mds/video/oracle1.mpg /mds/new_clips/oracle1.mpg
% mdscopy /mds/video/oracle1.mpi /mds/new_clips/oracle1.mpi
% vstagpatch -n "/mds/new_clips/oracle1.mpg" /mds/new_clips/oracle1.mpi
```

Example 2 This example modifies the tag file **oracle1.mpi** in the volume **video**, adding a comment and changing the bit rate to 2.048 Mbps:

```
% vstagpatch -d 'Introduction to Oracle7' -b 2048000 /mds/video/oracle1.mpi
```

Related Commands

vsgentag, **vsmkosf**, **vstag**, **vstagprint**

vstagprint

Use vstagprint to display the contents of a tag file from either an MDS file or a file on a host file system.

Syntax

`vstagprint [-c] [-h] [-P] [-R] [-s] [-T] [-V] input-filename`
where:

-c	certifies that tags are valid, but does not check against the content file itself. If **-s** is specified, **-c** is deactivated.
-h	prints usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	sets the specified resource descriptor. See "Using Resource Descriptors" on page 11-3 of this guide.
-s	prints only the header of the tag file without the entire contents.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	prints a version banner.
input -filename	specifies the file to be printed.

Usage Notes

Output of **vstagprint**:

```
 % vstagprint /mds.video/oracle1.mpi
Tag file version: 6.1
Current code is version 6.1, back-compatible to version 6.1
magic=aabbccdd (should be aabbccdd)
file describes content for: <Unavailable>
1 member files:
All rate control operations are allowed on this file
# 1. format=MPEG 1     transport=None     flags=1
file="/mds/video/oracle1.mpg" Size = 7167216
creation time of content: Aug  8 17:18:48
bitrate=2048000 bits/second
elapsed length=27996 milliseconds
```

```
presentation rate=1000
frames/sec(* 1000)=29970
57 tags in tag file.
width: 352, height: 240, pel aspect ratio (* 10000): 10950
MPEG1 Tagfile version: 1.1
Current MPEG1 code is version 1.1, back-compatible to 1.1
sequence header is 76 bytes starting at offset 0 of compression data
Tags for member file #1:
1:1, pos= 5315,   PTS- 16262  [002f0005a20b0000000000000000]
2:1, pos= 231383, PTS- 128875 [0b8b0034808d0000000a00000000]
3:1, pos= 471340, PTS- 241487 [0c14003f39930000000c00010000]
4:1, pos= 711650, PTS- 354100 [0c420055891d0000000f00010000]
5:1, pos= 951971, PTS- 466712 [0c7b00570b960000001100020000]
[etc.]
```

The output header tells you:

- the tag format version of the tag file and the versions accepted by the OVS. Different versions of the OVS may expect different tag format versions, so after you upgrade the OVS, this information tells whether you must regenerate your tag file. In this example, both the tag file format and the OVS tag format are version 6.1, so you need not regenerate.
- whether the input file you specified is a valid tag file. If the magic value does not match the text following it, either the input file is not a tag file or the input file is corrupted. In this example, the magic value “aabbccdd” matches the text following, so oracle1.mpi is a valid tag file.
- the video file described is an MPEG-1 system stream.
- the name and size of the content file to which the tag file points. In this example, the tag file is based on the content file **/mds/video/oracle1.mpg** which is 7,167,216 bytes in length.
- the bit rate of the associated content file. In this example, the bit rate of oracle1.mpg is 2,048,000 bps (2.048 Mbps).

> **Note:** The flags field can be ignored.

Each numbered line of output following the header describes a single tag point and has these fields:

Table 13–1 Tag points

integer1:*integer2*	tag number and the internal code for the type of the frame tagged.
pos	byte position in the content file where the tagged frame begins.
PTS or SCR	Presentation Time Stamp and System Clock Reference describe when the tagged frame appears as the video is playing.
[ ]	contains the information necessary to reposition the video to this tag point as it is playing.

Example

This example prints the header of the tag file video1.mpi located in the volume video:

```
 % vstagprint /mds/video/video1.mpi
Tag file version: 6.1
Current code is version 6.1, back-compatible to version 6.1
magic=aabbccdd (should be aabbccdd)
file describes content for: <Unavailable>
1 member files:
All rate control operations are allowed on this file
# 1. format=Raw Key Frame     transport=None     flags=1
file="/mds/video/video1.osf" Size = 15003788
creation time of content: Aug 11 11:11:47
bitrate=882437 bits/second
elapsed length=136021 milliseconds
presentation rate=1000
frames/sec(* 1000)=10000
272tags in tag file.
width: 240, height: 180, pel aspect ratio (* 10000): 10000
RKF Tagfile version: 1.1
Current RKF code is version 1.1, back-compatible to 1.0
Init data is 88bytes starting at offset 0 of compression data
```

14

Session-and-Circuit Service Utilities

The Session-and-Circuit Service utilities provide information about active client sessions and the resources allocated to them. These utilities include:

vscsmdir	The session and circuit list utility provides information about active client sessions and the resources allocated to each session.
vscsmkill	The session kill utility disconnects clients from Oracle Video Server.

vscsmdir

vscsmdir (circuit and session service directory utility) lists information about active client sessions, circuits, and resources allocated to them.

Syntax

```
vscsmdir [-a] [-s] [-S] [-t] [-T] [-c] [-C] [-m] [-M] [-e] [-E] [-b] [-B] [-x]
[-X] [-p] [-f] [-g] [-h] [-V] [-i clientDeviceId] [-R resource-statement]
[-P resource-file]
```

where:

-a provides session, circuit, channel, Media Net address, bit rate, and resource information for all active clients. The **-B**, **-C**, **-E**, **-R**, and **-S** options can be used to suppress specific information fields.

-b specifies that bit rate information be displayed

-B hides bit rate information when used in conjunction with **-a**.

-c specifies that circuit information be displayed

-C hides circuit information when used in conjunction with **-a**.

-e specifies that channel information be displayed

-E hides channel information when used in conjunction with **-a**.

-f displays the session and circuit service configuration file specified with the **-f** option of **vscsmsrv**.

-g lists the running session and circuit service configuration. The running configuration consists of the built in system default and, if applicable, the configuration file specified when **vscsmsrv** started.

-i specifies the client device ID

-m specifies that Media Net information be displayed

-M hides Media Net information when used in conjunction with **-a**.

-p	lists video pumps registered with the session and circuit service and any active channels.
-s	specifies that session information be displayed
-S	hides session information when used in conjunction with **-a**.
-t	displays the timestamp when the session was first allocated
-T	hides timestamp when used in conjunction with **-a**.
-x	specifies that resource information be displayed
-X	hides resource information when used in conjunction with **-a**.

Usage Notes

vscsmdir displays information about client sessions. **vscsmdir** can display information about all active or individual sessions.

The output of **vscsmdir** is separated by delimiter characters into the following syntax:

```
id [session info] [circuit info] [resource info]
```

where:

id	physical address of the client device.
session info	the number of circuits and resources associated with a session. A session is a collection of resources allocated by a particular client device. A client device may only have one active session at any given time on a given server.
circuit	the circuit represents a communications path from the client to the server. This is done by binding a *downstream* channel to a *upstream* channel and associating a Media Net address with that binding. Note that the channels contained within a circuit are not necessarily unique to the containing circuit.

resource a resource is a named (i.e. identified by a string) item that the session will hold an object reference (of type “any”) on behalf of the client or one of the services the client is accessing.

Examples

You can list all available information for active client connections with the **-a** option of **vscsmdir**. The **-a** option provides:

- client’s ID
- client’s physical address
- maximum bit rate a client’s video pump will provide over the communication channel
- communication channels between Oracle Video Server Command Reference and the client
- resources allocated on behalf of a client by Oracle Video Server Command Reference

This example displays two active clients:

```
% vscsmdir -a
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
ovc:oracle2-pc-418 27-Jan-1998 09:17:10.011 PST 2/0 [dupCD-P <1.0.41.667> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.21:5101 3100000 bps)]
```

About the Output

The concepts of the Session-and-Circuit Service are described in *Introducing Oracle Video Server*. You should be familiar with the concepts presented there before continuing in this section.

The syntax of the output provided by **-a** is of the form:

id [session_info] [circuit_info] [resource_info]

To better explain the syntax of the output, this example uses the following line of output to explain the different types of information presented, and where it appears in the output string. Output for the section being described appears in boldface text.

```
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

Client ID

```
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

Timestamp

The timestamp is the time the session was allocated to the client:

```
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

Session Information

Session information consists of the number of circuits allocated to a client and the number of resources used by the session. This example lists two allocated circuits but no resources:

```
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

Note that a client may have only one active session at a time with Oracle Video Server Command Reference.

Circuit Information

Circuit information consists of:

- properties associated with each circuit.
- Oracle Media Net address assigned to the circuit. The Oracle Media Net address of this circuit is 1.0.41.666.

You will find information on the properties of circuits in the section Properties of Circuits and Channels.

```
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

Channel Information

Channel information consists of:

- properties associated with each channel
- network protocol in use by the channel
- physical address of the client using the channel
- maximum bit rate the channel supports

You will find information on the properties of channels in the section Properties of Circuits and Channels.

```
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ]
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

Properties of Circuits and Channels

The syntax of the properties output is:

```
[{d|–} {u|–} {p|b|m} {C|–} {D|–} {I|–} {P|T}] [{+|-}]
```

where:

- downstream (d) or not (–)
- upstream (u) or not (–)
- pointcast (p), broadcast (b), or multicast (m)
- control data (C) or not (–)
- non-isochronous data (D) or not (–)
- isochronous data (I) or not (–)
- persistent (P) or transient (T)
- if the circuit or channel carries isochronous data, a plus sign (+) indicates that a data stream is bound to it. A minus sign (-) indicates that the data stream is not yet bound to the circuit or channel.

Based on these definitions, the example output line indicates that:

```
ovc:oracle-pc-389
```

The client ID is ovc:oracle-pc-389

```
27-Jan-1998 09:56:30.028 PST
```

The session was allocated on January 27, 1998 at 9:56 am Pacific Standard Time.

```
2/0
```

There are two circuits allocated, but no resources:

```
[dupCD-P <1.0.41.666> ]
```

The properties of the circuit are downstream (d), upstream (u), pointcast (p), control data (C), non-isochronous data (D), no isochronous data is present (–), and persistent (P). The Oracle Media Net address assigned to the circuit is 1.0.41.666.

```
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

The properties of the channel are, downstream (d), no upstream (-), pointcast (p), no control data (-), non-isochronous data is present (D), isochronous data is present (I), persistent data is present(P), and the data stream is bound to the channel (+).

The client is using the UDP protocol and the physical network address is 127.0.0.1:5100. The maximum bit rate that the channel can provide is 3100000 bps (3.10 Mbps). This value is based on the maximum bit rate the video pump allocated to this channel can provide.

Related Commands

vscsmsrv, **vscsmkill**

For Further Information

For further information on **vscsmdir** and the output it provides, refer to:

- Chapter 6, “Monitoring Clients”
- Chapter 10, “Configuring the Session-and-Circuit Service”

vscsmkill

vscsmkill disconnects clients from the OVS.

Syntax

```
vscsmkill [-h] [-V] [-T] [-R resource-statement] [-P resource-file]
client-id...
```

where:

-h	print usage information.
-P	specifies the resource file. See "Using Resource Descriptors" on page 11-3 of this guide.
-R	set the specified resource descriptor.
-T	enables verbose mode. Verbose mode returns information about actions performed by processes.
-V	print a version banner.
client-id	identifies one or more client devices to disconnect from the server.

Usage Notes

You may want to disconnect client devices if your Oracle Video Server system is running out of resources or if you plan to shut down the server. This utility disconnects a client device regardless of the operations it is currently performing, such as running an application or watching a video.

Use **vscsmdir** to obtain a listing of all active client physical addresses. Note that in addition to the client device's name, you must include the client's ID number.

Examples

Example 1 This example disconnects the PC client. For **clientID:video-pc-42.**

1. Obtain the ***client-id*** using the **vscsmdir** utility. This example uses the **-a** option to list clients connected to the server. The ***client-id*** portion of the output is in boldface text.

```
% vscsmdir -a
ovc:oracle-pc-389 27-Jan-1998 09:56:30.028 PST 2/0 [dupCD-P <1.0.41.666> ] \
[d-p-DIP+ (d-p--IP UDP:127.0.0.1:5100 3100000 bps)]
```

2. Specify that **vscsmkill** disconnect the client as shown:

```
% vscsmkill ovc:oracle-pc-389
```

Related Commands

vscsmdir

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

Y