HYPERtape
OpenMedia

Concepts and Facilities

MultiStream Systems



Copyright 1997 by MultiStream Systems.
Reproduction is prohibited without written permission of
MultiStream Systems. All rights reserved.

For further information, please read “Notice to the Reader”.

Please address any comments concerning this manual to your
local MultiStream support office or your software supplier.

REVISION HISTORY: ***HYPERtape OpenMedia Concepts and Facilities***

Revision date	Version/Level	Description
October 1997	3.0	Release

Notice to the Reader

This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced, transmitted or translated to another language in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without express written permission of MultiStream Systems. The information contained in this document is subject to change without notice.

Copyright © 1997, MultiStream Systems

This document refers to numerous products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. The references are made for informational purposes only; no consent is implied.

HYPERtape and HYPERtape OpenMedia are registered trademarks of MultiStream Systems Inc.

About this Manual

This manual provides an overview of HYPERtape OpenMedia, explaining what it is and what it does. The manual is intended for anyone who needs an introduction to OpenMedia, from technical decision makers and system managers who are assessing the product, to software engineers who are developing applications that will integrate with it.

How to use this manual

The purpose of the manual is to introduce the concepts and features of OpenMedia. You should read it through before starting to use the product. There are three chapters:

Chapter 1 **Introducing HYPERtape OpenMedia** defines the product, lists its main features and describes its design principles.

Chapter 2 **HYPERtape OpenMedia Model** explains how items such as tapes, jukeboxes and locations are defined as objects, and how these objects relate to each other in the object model.

Chapter 3 **Using HYPERtape OpenMedia** summarizes the functions of the server and client software and introduces the Application Program Interface (API), Graphical User Interface (GUI) and Command Line Interface (CLI).

Further reading

The following documentation is available when you are familiar with the basics of OpenMedia:

- *HYPERtape OpenMedia System Manager Guide*

  This manual is intended for the system manager who is an end user of OpenMedia. It explains how to install the product and how to manipulate the OpenMedia database using the Graphical User Interface and Command Line Interface.

- *HYPERtape OpenMedia Application Program Interface*

  This reference manual is for application programmers. The Application Program Interface is largely self documenting, so this manual concentrates on concepts, techniques and examples.

These documents are supplemented by the comprehensive help text supplied within the product.

Need help?

OpenMedia is designed to be easy to use and operate. However, if you have questions about the product or encounter a problem with the software, please contact your software supplier, or the MultiStream support office nearest you, for assistance.

Before reporting a software problem, please record the commands that preceded the error and the exact error message that appeared. With this information on hand, the support staff are better able to help you with the problem.

Contents

About this Manual

1 Introducing HYPERtape OpenMedia

What is HYPERtape OpenMedia? 1-1
What does HYPERtape OpenMedia do? 1-1
HYPERtape OpenMedia objects 1-2
How is HYPERtape OpenMedia used? 1-2
How does HYPERtape OpenMedia work? 1-2
Platforms 1-3
Design principles of HYPERtape OpenMedia 1-4

2 HYPERtape OpenMedia Model

Objects 2-1
HYPERtape OpenMedia object model 2-2
Media operations 2-3
Device capabilities 2-5
Management information 2-5
Stored data 2-6
User administration 2-7
Other relationships in the model 2-8
Object classification 2-9
Media objects 2-9
Administration objects 2-10

3 Using HYPERtape OpenMedia

HYPERtape OpenMedia server 3-1
Startup and shutdown 3-2
Security and privilege 3-2
Error and disaster recovery 3-2
Graphical User Interface (GUI) 3-2
Command Line Interface (CLI) 3-3
Administrative Utilities and Ad Hoc jobs 3-4
Application Program Interface 3-4
Object representation 3-5
Object manipulation 3-5
Writing client programs 3-6

Index

1 Introducing HYPERtape OpenMedia

What is HYPERtape OpenMedia?

OpenMedia is a software tool for managing storage media across large and small enterprises. It has been developed by MultiStream Systems, as part of their commitment to provide industry-leading network data management products to customers worldwide.

OpenMedia is one of a new generation of software products, which fulfill customers' needs for scaleable and flexible solutions in the areas of multiplatform network backup, network attached data storage, and multimedia Intranet and Internet communications.

What does HYPERtape OpenMedia do?

The main facilities offered by OpenMedia are:

- Volume management
- Saveset management
- Device management
- Robotics or human operator control
- User and host administration and authorization
- System policy application

HYPERtape OpenMedia objects

OpenMedia maintains a database of ***objects***, which can be physical items such as the volumes and host systems in your enterprise, or abstract entities such as media pools and user privileges.

Objects and the defined relationships between them are encompassed in the OpenMedia ***model***. OpenMedia provides you with the tools you need to describe your own organization, and to monitor, control and automate your storage management.

How is HYPERtape OpenMedia used?

OpenMedia is flexible and adaptable, which makes it suitable for use in a wide variety of organizations. It can stand alone as a media manager, work in concert with other MultiStream HYPERtape components, or integrate with other storage applications.

It is designed to be easy to integrate with both robot managers and tape writing products such as backup or HSM applications, including HYPERtape Enterprise Backup.

When integrated with a robot manager, OpenMedia automatically chooses suitable tapes, but calls the robot manager to issue load and unload commands on the chosen tapes. OpenMedia monitors and records the location and status of the robot's tapes and devices when the robot loads, unloads and relocates the tapes.

The Application Program Interface (API) is designed to help Value Added Developers (VADs) and Original Equipment Manufacturers (OEMs) integrate OpenMedia with their own products.

How does HYPERtape OpenMedia work?

At the heart of OpenMedia is a database with an ad hoc query mechanism. The software is implemented using a client/server architecture. The server does most of the work of OpenMedia, in response to requests generated by the clients.

Most client / server applications are static: they require manual intervention when a client or server subsystem is moved to another host. By contrast, OpenMedia uses a ***dynamic*** service broker. This means that clients ask for a service rather than a particular server, and are routed to the server currently offering that service.

Clients can use services on the network without knowing where the server is running, so there is no need to update client and peer processes with server location information when a server process is moved to another host. After the server starts, it registers the service(s) it offers with the service broker.

Multiple service brokers can be defined to provide redundancy, so that clients can always find a broker and then locate the service. In addition, you do not have to reconfigure all the clients when a server is temporarily moved.

A number of clients are provided as part of OpenMedia. These include a Graphical User Interface (GUI) and a Command Line Interface (CLI) for writing scripts. Utility components for administering the system are also included within the product.

OpenMedia's operations can be divided into two types: media and administration. For example, the management of tapes is media related, whilst the management of users and their privileges is administration related. This division of work runs through the design.

Platforms

Clients are supported for many operating systems. See the *HYPERtape OpenMedia Software Product Description* for a full list.

OpenMedia works in ***homogeneous*** environments, which use the same operating system throughout, and ***heterogeneous*** environments, in which different operating systems are used by different subsystems.

When new releases of the API and server executable are made, users are generally unable to upgrade all parts of a distributed product at the same time. For this reason, provision has been made for a client and server to work correctly whilst they are one release apart from each other. For example, a Version 3 client can communicate with a Version 4 server and a Version 4 client can communicate with a Version 3 server.

Design principles of HYPERtape OpenMedia

OpenMedia meets the following stringent design criteria:

- ***Server/client architecture***
- ***Standard look-and-feel***
- ***Cross-platform capability***
- ***Common view interfaces***
- ***Extendibility***
- ***Scaleability***
- ***Flexibility and configurability***

Server/client architecture

OpenMedia is implemented using client/server processing with a dynamic service broker. After the server starts, it registers the services it offers with all service brokers.

Standard look-and-feel

A standard MultiStream look-and-feel is implemented in OpenMedia, although this can be customized by third parties (VADs and OEMs) who wish to implement their own look-and-feel.

Cross-platform capability

OpenMedia is designed to be inherently portable. The first release includes support for Windows NT (Intel) and a number of UNIX platforms.

Common view interfaces

OpenMedia interfaces provide a common view of the data management environment. When running the user interfaces from any client workstation on the network, users can manage and administer data management tasks across the enterprise.

This is true for both homogeneous and heterogeneous environments. The operating systems and network protocols used when performing data management tasks are irrelevant from the user's point of view. The user interface is the same, regardless of host.

This provides the flexibility necessary for system managers and administrators to manage their site's storage from any host, providing the GUI software is installed. [1]

Extendibility

OpenMedia is easily extendible to new platforms, devices and database management systems. Most components are platform independent, so there is no need to rewrite for new target platforms. This allows for very quick porting to new platforms.

Components are also highly modular, so changing one does not generally affect others. All server components provide a published set of API calls, allowing external processes to make requests to the server. This is equally true for MultiStream software or third party software products utilizing the servers.

Scaleability

OpenMedia is scaleable for use at large or small customer sites. It can be introduced into a small organization and grow with the storage management needs of the organization. For example, new jukeboxes and backup clients can easily be added.

Conversely, it can be introduced into large sites that have large amounts of data distributed across a network.

Flexibility and configurability

OpenMedia is flexible and configurable. Through the interfaces provided, the user or support personnel can set various system policy / environment variables that allow fine tuning of the product for a particular site.

---

[1] The GUI can accept look-and-feel instructions to behave as Motif, Open-Look or Windows.

Whenever possible, reasonable defaults are set during installation, so that the product is ready to work straight away.

2 HYPERtape OpenMedia Model

Objects

In order to exploit the full power of OpenMedia, it is important to understand about objects and their manipulation. Objects represent items in the enterprise. They can represent physical items such as tapes, devices and media locations, or abstract items such as media types, media pools and system policies. They can also represent certain links, or *intersections*, between other objects. For example, both the media types that can be read by a device and the devices that are accessible to a host are identified by OpenMedia objects.

There can be many instances of each object in the database. For example, *Vault1* and *Desk_Drawer* might be two of the media locations in the database; *Exabyte* and *DAT* might be two of the media types.

Many objects have titles that make their purpose obvious. For example, a media location is an object that represents a place where media is located, such as a vault or computer room; a host is an object that represents a host system.

HYPERtape OpenMedia object model

The OpenMedia object model describes the entire enterprise by defining the ways that objects relate to each other. For example, tapes reside in media locations, which reside in a locality. Some relationships are one-to-many: one media location can contain several tapes, but a tape can reside in only one media location at a time. Other relationships are many-to-many: one device can read several media types and one media type can be read by several devices.

Many-to-many relationships can be ranked so that some related objects have precedence over others. For example, a media location has most and least preferred devices. To reduce the time operators spend walking between the devices and the tape store, you could give higher preference to the device closest to the tape store.

Relationships can be optional or mandatory. For example, a media location does not have to reside within a jukebox, but a tape must reside in a media location. In most cases, mandatory relationships are used to ensure that all objects are related directly or indirectly to a locality.

In the future, a hierarchy of localities may be introduced, with a domain locality at the top and regional and neighborhood localities below, as shown in Figure 2.1.

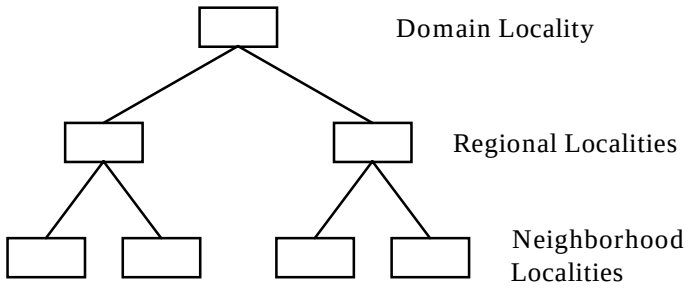


Figure 2.1 The hierarchy of localities

At present, and throughout this document, the term locality refers to a single object that is both a domain and a neighborhood locality.

The rest of this section describes the most important relationships in the model. For simplicity, the model is broken down as follows:

- Media operations
- Stored data
- General administration
- User administration

For a full list and brief description of all the media and administration objects, see the section on *Object classification* at the end of this chapter.

The major relationships in these areas are represented in a number of diagrams, which use the following symbols:

Symbol	Meaning
Host *Jupiter*	Object. The object is shown in regular type. Example names are shown below, in italics.
- - - - - - - - - ->	Optional one-to-many relationship. The many point to the one.
——————>	Mandatory one-to-many relationship. The many point to the one.
<- - - - - - - - ->	Optional many-to-many relationship.
<——————>	Mandatory many-to-many relationship.

The diagrams contain examples of typical names for objects. The examples show an object model, which is constructed to describe how tapes, devices, media locations and so on are related to each other and to a locality with the name *London_Data_Center*.

Media operations

Figure 2.2 shows the relationships between objects involved in media operations by describing how a single manual tape drive might be modeled. In the figure, the device *DLT001* is connected to the host *Jupiter* and is one of the drives controlled by the jukebox *Operators*, who can mount any tape in the media location *Computer_Room*. This means that *Operators* can only mount a tape if it is in the *Computer_Room*.

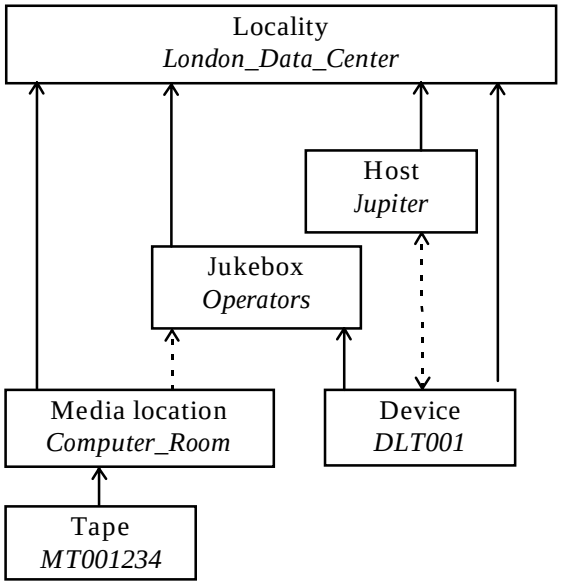


Figure 2.2 Media operation relationships

Note that:

- Media locations, jukeboxes, hosts and devices all relate to a locality. In the diagram, the *Jupiter* host, *DLT001* device, *Computer_Room* media location and *Operators* jukebox all reside in the *London_Data_Center* locality.

  Jukeboxes determine whether tapes in given media locations can be loaded on given devices. They can represent robots capable of loading and unloading devices automatically, or media location and device groupings in which human operators carry out manual loading and unloading operations. In robotic jukeboxes, the media location (normally just one) is the collection of slots within the robot.

- A media location is related to a jukebox if the tapes it contains can be loaded on a device. An offsite media location will not be related to a jukebox if tapes cannot be loaded there. In the *London_Data_Center*, the *Computer_Room* media location is related to the *Operators* jukebox.

- A device is always related to a jukebox. In the *London_Data_Center*, the *DLT001* device is related to the *Operators* jukebox.

- A device is related to any hosts capable of accessing it. In the *London_Data_Center*, device *DLT001* can be accessed by host *Jupiter*.

Device capabilities

Figure 2.3 shows the relationships between objects that relate to device capability. The tape *MT001234* is a *DLTTapeIV* cartridge (the physical type), initialized to *DLT70GBCompressed* (the media type), and the device *DLT001* can both read and write *DLT70GBCompressed* tapes.

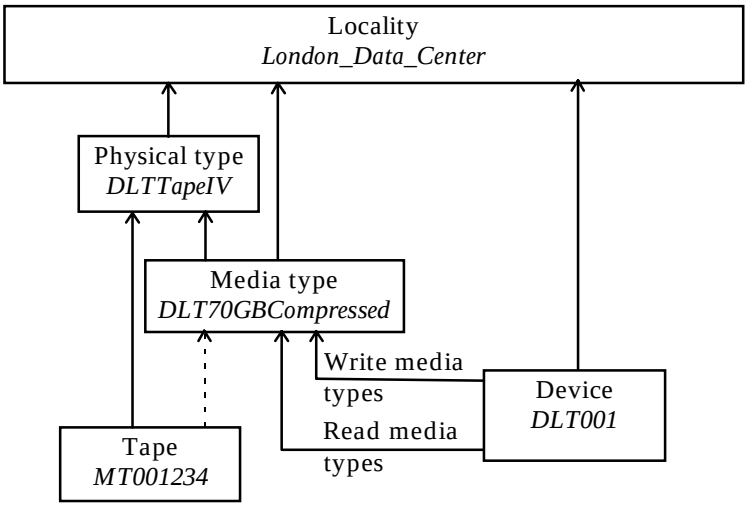


Figure 2.3 General administration relationships

Note that:

- Media types, physical types and devices all relate to a locality. The diagram shows objects of these types residing in the *London_Data_Center* locality.
- A media type is always associated with a physical type. For example, *DLT70GBCompressed* is associated with *DLTTapeIV* in the *London_Data_Center*.
- A tape always has a physical type. If it is not BLANK, it will also have a current media type. In the *London_Data_Center*, *MT001234* is a *70GBCompressed DLTTapeIV* tape.

Management information

Figure 2.4 shows the relationships between objects involved in the storage of management information. The public media pool *Full_Backups* draws tapes from the scratch media pool *Scratch* when it needs them.

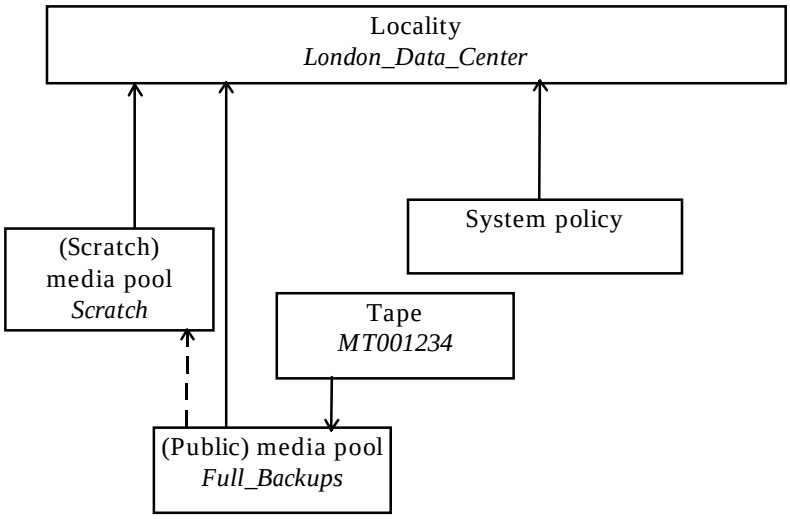


Figure 2.4 General administration relationships

System policy objects represent defaults set for the whole locality.

Note that:

- Media pools and system policies relate to a locality. The diagram shows objects of these types residing in the *London_Data_Center* locality.
- A tape always belongs to a media pool. In the *London_Data_Center*, the tape *MT001234* resides in the *Full_Backups* media pool.

Stored data

Figure 2.5 shows the relationships between tapes and the data stored on them. The saveset *Full_Backup_of_Jupiter* is stored on a segment on tape *MT001234*. The same saveset might have needed more than one tape, in which case there would be multiple segments relating the saveset to the set of tapes on which it resides.

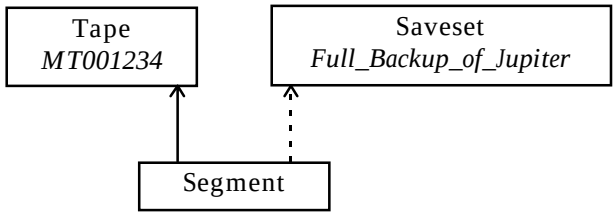


Figure 2.5 Stored data relationships

Note that:

- A segment represents the portion of a certain saveset on a certain tape. Segments do not have names.
- A saveset can be apportioned across several tapes.
- A tape can contain portions of several savesets.
- When a saveset is removed, its segments remain associated with tapes until the tapes are scratched.

User administration

Figure 2.6 shows the relationships between objects involved in user administration. The user *John_Smith* has *Operator* privilege and, if he logs in as *John* on host *Jupiter*, or as *JSmith* on host *Saturn*, he will not need a password.

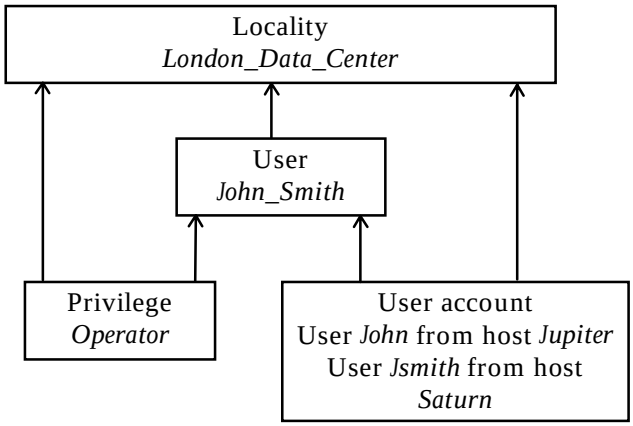


Figure 2.6 User administration relationships

Note that:

- Users, user accounts and privileges all relate to a locality.
- A user account maps a host's native user name to an OpenMedia user name. In the *London_Data_Center*, the user account objects map the *Jupiter* user named *John* and the *Saturn* user named *JSmith* to the *John_Smith* user.

A privilege bestows rights to a user within a locality. In the *London_Data_Center*, user *John_Smith* has *Operator* privilege.

Other relationships in the model

Relationships not shown in the diagrams include the following:

- A media location can have preferred devices. This is an optional, ranked, many-to-many relationship, where devices are ranked in order of preference for each media location.

  In an operator controlled jukebox, this can be used to reduce the time human operators spend walking between the device and the tape store. In robotic jukeboxes such as a Silo with multiple media locations and devices, it can be used to avoid unnecessary pass through of the tapes.

- Media locations in robotic jukeboxes can identify a *dropped* media location via an optional one-to-many relationship. If a tape ceases to be reported by a jukebox, the location of the tape is no longer known. In this case, its media location reference can be altered to the dropped media location. For example, the media location *Robot1* might identify a dropped media location of *Table_Beside_Robot1*. If a tape in the jukebox ceases to be reported, its media location can be altered from *Robot1* to *Table_Beside_Robot1*.

- Media pools can identify default onsite and offsite media locations for the tapes belonging to them, via optional one-to-many relationships.

- Tapes can identify their onsite and offsite media locations in addition to their current media location, via optional one-to-many relationships.

- A saveset can identify the previous saveset to which it relates, via optional one-to-many relationships.

- A host can optionally run a service broker for a number of localities. Every locality must have at least one broker host. This is a many-to-many relationship.

- Devices and jukeboxes can be associated with a control host. This is the host upon which their robotic control can be executed.

Object classification

OpenMedia objects are classified as follows:

Classification	Meaning
Media	Object is a representation of a media item
Administration	Object is a representation of an administration item
Auxiliary	Object is a control structure
Request	Object is a request to a server
Response	Object is a response from a server
Database	Object is maintained in the database
Volatile	Object is maintained in server memory, not in the database
Prompt	Object is a prompt to a human operator or robotic jukebox controller
Reply	Object is a reply to a prompt

The next two sections list the media and administration objects. The requests initiated by clients, and the responses to requests generated by the server are also objects. For lists of these objects, see the manual *HYPERtape OpenMedia Application Program Interface.*

Media objects

The following table lists the media objects and indicates whether they are volatile (V) or held in the database (D):

Object	D/V	Brief description
Media location	D	Location for tapes
Jukebox	D	Relates media locations to devices
Device	D	Item on which tapes can be loaded
Tape	D	Unit of recording media
Saveset	D	Collection of files on tape(s)
Segment	D	Portion of a saveset on a tape
Reserved device	V	Represents a reserved device
Reserved tape	V	Represents a reserved tape
Opened saveset	V	Represents an opened saveset
Opened segment	V	Represents an opened segment

Object	D/V	Brief description
Active stream	V	Represents an opened saveset's active segment stream
Device status	V	Holds the current status of a device
Tape status	V	Holds the current status of a tape
MType read by device†	D	Device's readable media types
MType write by device†	D	Device's writeable media types
Device on host †	D	Host's accessible devices
Device pref by location†	D	Media location's preferred devices

† These objects are *intersection* objects representing many-to-many relationships between other objects.

Administration objects

The following table lists the administration objects and indicates whether they are volatile (V) or held in the database (D):

Object	D/V	Brief description
Media pool	D	Groups tapes and supplies defaults
Media type	D	Qualifies devices for tapes
Physical type	D	Tape's physical type or shape
Locality	D	Site within the enterprise
Host	D	Computer within the enterprise
User	D	OpenMedia user name
User account	D	Host native user name
Privilege	D	User privileges in a locality
System policy	D	Configuration or preference
Host brokering †	D	Which hosts are brokered by which hosts

† This object is an *intersection* object representing many-to-many relationships between other objects.

3
Using HYPERtape OpenMedia

HYPERtape OpenMedia server

The operation of OpenMedia is undertaken by server software. Several servers may be installed in an enterprise, but they do not share any knowledge of that enterprise. Each server provides the media management for a particular locality. If multiple servers are installed, clients can ask for disparate services and may be connected to different servers.

The initial connection between clients and servers is established through a service broker: clients ask for a service rather than a particular server and are connected to any server offering that service. The client then uses the API to create request objects. After executing a request, the server returns a response object.

Clients log in to the server by *password accreditation* or *native accreditation*. In password accreditation, the user specifies an OpenMedia user name and password. In native accreditation, the user logs in from a host whose host/account combination is stored as a user account in the OpenMedia database.

✔ ***Note*** *For security, passwords are transmitted on the network in an encrypted form.*

Startup and shutdown

Servers are normally started by an *Agent* when their host is booted, then run continuously. When a server starts, it assesses the services it is to offer. In particular, it determines which locality it is to serve. A client can shut down a server by issuing a shutdown request. For further details, see the manual *HYPERtape OpenMedia System Manager Guide.*

Security and privilege

The server maintains a record of known users and privileges. The client establishes the security credentials of a user by requesting accreditation from the server. Recognizing a known user is not in itself sufficient to allow access to OpenMedia services. Users are granted privileges in localities and the user's privilege within a locality is checked before services are provided.

Error and disaster recovery

Facilities are provided for the following:

- Retention of connection data if the connection of the server with a client is lost
- Recovery from abnormal termination of the server
- Database maintenance

For full details of these facilities, see the manual *HYPERtape OpenMedia System Manager Guide.*

Graphical User Interface (GUI)

The Graphical User Interface (GUI) is a client application that allows manipulation of objects in the database from a host workstation. For example, a user can interrogate the database to discover which tapes are past their expiry dates, then scratch them ready for reuse, or see which tapes are due to be sent to an offsite location.

The purpose of the GUI is to allow system managers and operators to manage, interrogate and change data in the OpenMedia database. It provides facilities for:

- Looking up, updating, adding and deleting database objects
- Looking up volatile objects
- Shutting down the server
- Responding to prompts

The GUI can accept look-and-feel instructions to behave as Motif, Open-Look or Windows.For further details, see the manual *HYPERtape OpenMedia System Manager Guide* and the comprehensive context-sensitive help included with the GUI.

Command Line Interface (CLI)

The Command Line Interface (CLI) is a client application that provides a way of issuing server requests directly from the command line. Its primary purpose is to allow the development of scripts for managing media and integrating OpenMedia with other applications, but it can also be used interactively for example by dial up telnet applications for remote support.

The CLI is a set of commands with associated requests such as to reserve a tape. Supported operations include:

- Looking up, updating, adding and deleting database objects
- Looking up volatile objects
- Shutting down the server
- Loading, reserving, releasing, unloading and looking up the status of devices
- Looking up the status of reserved devices
- Manipulating and reserving tapes
- Opening and closing savesets
- Opening and closing segments
- Writing and running reports

For further details, see the manual *HYPERtape OpenMedia System Manager Guide*, and the comprehensive help included with the CLI.

Administrative Utilities and Ad Hoc jobs

A number of administrative utilities and ad hoc jobs are provided as OpenMedia clients. For a full list, see the manual *HYPERtape OpenMedia System Manager Guide*.

Application Program Interface

The API allows client applications to interact directly with the server. The API header files, libraries and documentation are distributed as a separate product option. The API is used internally by the GUI, CLI, startup agent and jukebox controller components of OpenMedia. You do not need to purchase the API to use these components: you need it only if you want to write your own client applications.

Although this section relates to the optional API module, it may help readers with some technical knowledge to understand how OpenMedia works.

Client programs developed use the API to manipulate OpenMedia objects. The API consists of C header-include files and libraries. It has a strict naming convention, which groups objects with their functions and other declarations into component *subsystems*.

One subsystem contains all the administration and media objects. This subsystem is called *admed* and appears in client applications as *zadmed*, as all API identifiers start with z to distinguish API code from other code.

Subsystems containing auxiliary objects include the linked list *zlist* and selection criteria *zcrit*. Others offer functions; these include the administration / media client *zamclient* and the cryptographic checksum *zcrypt*.

For further information about the API, see the manual *HYPERtape OpenMedia Application Program Interface*.

Object representation

Each object consists of fields, some of which are simple integers or characters, whilst others are structured like objects and called *sub-objects*.

A one-to-many relationship between objects is defined by a *uuid* field, which holds the Universally Unique Identifier (uuid) of a single object. Each of the many objects to which a single object is related hold the uuid of that object. For example, a tape must be in a media location, so the tape's UuidOfMediaLocation field must contain the uuid of a media location.

Using the uuid of an object to relate it to other objects allows you to change names (rename an object) without affecting any of the relationships. The GUI and CLI normally hide the uuid, allowing the user to manipulate and relate objects by name.

Many-to-many relationships are represented in the database by intersection objects. Adding or deleting an intersection object adds or deletes a relationship. For example, to indicate that a device is available to a host, you simply add a device on host object. The intersection objects also hold any ranking information for the relationship.

Object manipulation

The API provides a basic set of functions for creating, deleting and manipulating objects and for manipulating sub-objects. For example, there is a Make function for every object, which allows you to create objects.

When functions are declared, the name of the subsystem is followed by:

- A verb describing its action
- The name of the object or sub-object concerned
- A field name, if applicable

So, for example, the function zadmedSetTapeComment sets the Comment field of a tape to a specified text string. Prototypes for all OpenMedia functions are supplied with the API.

Writing client programs

Client programs use the API to create request objects for execution by the server. After servicing a request, the server returns a response object. For example, the first request from a client is usually to establish the security credentials of the client's user with the server.

As a request is an object and every object has a Make function, the simplest way to create a request is to Make a request object. There are many types of request, and they are grouped by functionality as follows:

- General requests
  - Server management
  - Request management
  - Database Add, Update, Delete
  - General purpose database and server lookup
  - Change registration
  - Security
- Media requests
  - Devices
  - Tapes
  - Reserved tapes and devices
  - Opened savesets
  - Prompts

Index

A

Accreditation 3-2
Active stream 2-10
Ad Hoc jobs 3-4
Admed 3-4
Administration objects 2-9, 2-10
Administration / media client 3-4
Administrative utilities 1-3, 3-4
Agent 3-2
API 1-2, 1-5, 3-1, 3-4, 3-5
Application Program Interface *See* API
Auxiliary objects 2-9, 3-4

B

BLANK tapes 2-5
Broker host 2-8

C

Change registration 3-6
CLI 1-3, 3-3
Client application 1-3, 3-1, 3-4, 3-6
Command Line Interface *See* CLI
Common view interfaces 1-4
Configurability 1-4, 1-5
Credentials 3-2
Cross-platform capability 1-4
Cryptographic checksum subsystem 3-4

D

Database 1-2
Database Add, Update, Delete 3-6
Database maintenance 3-2
Database objects 2-9, 2-10, 3-3
Default onsite and offsite media locations 2-8
Design principles 1-4
Device 2-4, 2-5, 2-8, 2-9, 3-3, 3-5
Device management 1-1
Device on host 2-10
Device pref by location 2-10
Device status 2-10
Disaster recovery 3-2
Dropped media location 2-8
Dynamic service-broker lookup 1-3, 1-4

E

Enterprise Backup 1-2
Error recovery 3-2
Extendibility 1-4, 1-5

F

Facilities 1-1
Flexibility 1-4, 1-5

G

General administration 2-3, 2-5
General purpose database and server lookup 3-6
General requests 3-6
Graphical User Interface *See* GUI
GUI 1-3, 3-2, 3-3
GUI facilities 3-3

H
Heterogeneous environment 1-3, 1-5
Homogeneous environment 1-3, 1-5
Host 2-4, 2-8, 2-10
Host administration 1-1
Host brokering 2-10
Human operator 1-1, 2-4
HYPERtape Enterprise Backup 1-2
HYPERtape OpenMedia Introduction 1-1

I
Intersection objects 2-1, 2-10, 3-5
Introducing HYPERtape OpenMedia 1-1

J
Jukebox 2-4, 2-8, 2-9

L
Linked list subsystem 3-4
Locality 2-4, 2-5, 2-6, 2-7, 2-10, 3-1, 3-2
Look-and-feel 1-4

M
Make function 3-5
Mandatory relationship 2-2
Manipulating and reserving tapes 3-3
Many-to-many relationship 2-2, 2-8, 2-10, 3-5
Media location 2-4, 2-8, 2-9
Media objects 2-9
Media operations 2-3
Media pool 2-6, 2-8, 2-10
Media requests 3-6
Media type 2-5, 2-10
Model 1-2, 2-1, 2-2
Motif 3-3
MType read by device 2-10
MType write by device 2-10

N
New releases 1-3

O
Object 1-2, 2-1
Object classification 2-9
Object manipulation 3-5
Object model *See* Model
Object representation 3-5
Offsite media location 2-4, 2-8
One-to-many relationship 2-2, 2-8
Onsite media location 2-8
Opened saveset 2-9, 3-6
Opened segment 2-9
Opening and closing savesets 3-3
Opening and closing segments 3-3
Open-Look 3-3
Optional relationship 2-2

P
Physical type 2-5, 2-10
Platforms 1-3
Porting 1-5
Preferred device 2-8
Privilege 2-7, 2-10, 3-2
Prompt objects 2-9
Prompts 3-6

R
Ranked many-to-many relationship 2-2, 2-8
Recovery 3-2
Relationships between objects 2-2
Reply objects 2-9
Request management 3-6
Request objects 2-9, 3-6
Reserved device 2-9, 3-3, 3-6
Reserved tape 2-9, 3-6
Responding to prompts 3-3
Response objects 2-9
Retention of connection data 3-2
Robotics 1-1, 2-4

S
Saveset 2-7, 2-8, 2-9, 3-3
Saveset management 1-1
Scaleability 1-4, 1-5
Security 3-2, 3-6
Segment 2-7, 2-9, 3-3
Selection criteria subsystem 3-4
Server 1-2, 3-1, 3-6
Server management 3-6
Server/client architecture 1-4
Service-broker 1-3
Shutdown 3-2
Standard look-and-feel 1-4
Startup 3-2
Stored data 2-3, 2-6

Subsystem 3-4
System policy 1-1, 1-5, 2-10

T
Tape 2-4, 2-5, 2-7, 2-8, 2-9, 3-3, 3-6
Tape status 2-10

U
Universally Unique Identifier *See* Uuid
User 2-7, 2-10, 3-2
User account 2-7, 2-10
User administration 1-1, 2-3, 2-7
Using HYPERtape OpenMedia 3-1
Uuid 3-5

V
Volatile objects 2-9, 2-10, 3-3
Volume management 1-1

W
Windows 3-3
Writing client programs 3-6

Z
zadmed 3-4
zamclient 3-4
zcrit 3-4
zcrypt 3-4
zlist 3-4