Extended High Performance Data Mover

Operator and System Programmer's Guide

E25075-01

Version 6.2

Revision 01

Operator and System Programmer's Guide

About this Book

Scope

This manual describes basic concepts, startup, operation, optimization and some basic operation scenarios for the Extended High Performance Data Mover (ExHPDM) software product. ExHPDM is a Storage Technology Corporation (StorageTek) software product that is a client and server-based solution that manages the I/O from application programs and interleaves it, at the block level, onto the media. ExHPDM addresses the issues associated with disk backup and recovery.

Intended Audience

This manual is intended primarily for data center operators and systems programmers responsible for operating and maintaining ExHPDM software. Computer system administrators may also find information contained in this guide to be useful for reviewing and understanding ExHPDM concepts.

Users responsible for installation and maintenance of ExHPDM software involving the technical details should be familiar with the following software topics:

- MVS operating system
- System Modification Program (SMP/E)

About the Software

Extended High Performance Data Mover (ExHPDM) Version 6.2 is supported by this manual.

Table of Contents

Scope . . . 3
Intended Audience . . . 3
About the Software . . . 3

Table of Contents . . . 5

Chapter 1. Introduction . . . 1

Overview . . . 1
ExHPDM Description . . . 1
DFSMSdss and FDR in ExHPDM . . . 2
ExHPDM Concepts . . . 3
Administration Utility—SOVADMN . . . 3
DFSMSdss Interface—SOVDSSU . . . 4
ExHPDM Client . . . 4
Connection . . . 5
Expected Flow Increase . . . 5
Flow Limit . . . 5
ExHPDM MVS Server . . . 5
Stream . . . 6
Stream Class . . . 7
Stream Device . . . 7
Stream File . . . 7
Stream Management . . . 7
Stream Task . . . 7
ExHPDM Database . . . 7
ExHPDM Journal Data Set . . . 8
Using Non–ExHPDM Utilities for the ExHPDM Database . . . 8
ExHPDM Startup Parameter File . . . 8
ExHPDM and Z/OS 1.9 . . . 8

Chapter 2. License Key . . . 9

ExHPDM License Key . . . 9
Serial Number . . . 9
Obtaining a Permanent License Key . . . 9
Obtaining an Emergency License Key . . . 10
License Key Validation . . . 10
. . . 11
. . . 11

Chapter 3. Operation . . . 13

Overview . . . 13
ExHPDM Processing . . . 14
ExHPDM Clients . . . 14
DFSMSdss . . . 15

FDR 22
ExHPDM Administration Utility (SOVADMN) 33
Examples 33
Database Processing 34
Disaster Recovery of Client Data Sets 34
Directing Utilities through the SYSIN File 34
ExHPDM Server 34
Identify a Client Connection for ExHPDM Processing 36
Identify the Performance Attributes for the Client Connection 37
Identify the Management Attributes for the Client Data Set 38
Identify the Stream and Stream Task for the Client Connection 41
Identify an Input or Output Device to Service the Stream Task 42
The Relationship of Stream Parameter Definitions 43
Abnormal Termination of Client Connections 45
System Security, Authorization, and Verification 46
Using ExHPDM with Your Tape Management System 48
Using ExHPDM in a Disaster Recovery Process 53
Reinstating a Corrupted ExHPDM Database 55
The Consumption of Address Space IDs (ASID) in MVS 57
ExHPDM Diagnostic Facilities 57
ExHPDM and MVS Time Changes 59
ExHPDM's use of the MVS TOD Clock 59
MVS Time Changes 60

Chapter 4. Optimizing ExHPDM Performance 61

Overview 61
Load Balancing 62
Examples 64

Chapter 5. ExHPDM Scenarios 71

ExHPDM Scenarios 71
Basic ExHPDM Server Started Task JCL 71
STARTING the ExHPDM Server 71
Basic ExHPDM SOVPRMxx Parameter File Used for Startup 71
SMC Example 73
DFSMSdss Examples 73
Using Multiple ExHPDM Servers from a Single Job 78
FDR Examples 79
Listing the ExHPDM Database 85
Deleting from the ExHPDM Database 85
Deleting Expired Client Data Sets from the ExHPDM Database 86
Using SCANSTREAMFILE from SOVADMN 86

Chapter 6. . . . . ExHP-DM Startup Parameter File 89

Overview 89
Defining the Startup Parameter File 89

Sharing Parameter File Definitions Using System Symbols 92
Customizing the ExHPDM Parameter File 93
ACTIVATE Keyword 95
Syntax 95
Keyword Name 95
Parameters 95
ACTIVATE Keyword Examples 97
CLASS Keyword 99
Syntax 99
Keyword Name 99
Parameters 99
CLASS Keyword Examples 100
DATABASE Keyword 102
Syntax. 102
Keyword Name 102
Parameters 103
DATABASE Keyword Examples 105
DEVICE Keyword 107
Syntax 107
Keyword Name 108
Parameters 108
DEVICE Keyword Examples 114
LKEYINFO Keyword 115
Syntax 115
Keyword 115
Parameters 115
LKEYINFO Example 116
LOGFILE Keyword 117
Syntax 117
Keyword Name 117
Parameters 117
LOGFILE Keyword Example 118
MANAGEMENT Keyword 121
Syntax 121
Keyword Name 121
Parameters 121
MANAGEMENT Keyword Examples 123
MONITOR Keyword 124
Syntax 124
Keyword Name 124
Parameters 125
MONITOR Keyword Example 126
PREFIX Keyword 128
Syntax 128
Keyword Name 128
Parameters 128

PREFIX Keyword Examples . . . 128
REQUEST Keyword . . . 130
Syntax . . . 130
Keyword Name . . . 130
Parameters . . . 130
REQUEST Keyword Examples . . . 131
SAF Keyword . . . 133
Syntax . . . 133
Keyword Name . . . 133
Parameters . . . 133
SAF Keyword Example . . . 133
SELECT Keyword . . . 134
Syntax . . . 135
Keyword Name . . . 135
Parameters . . . 135
SELECT Keyword Examples . . . 138
SMF Keyword . . . 140
Syntax . . . 140
Keyword Name . . . 140
Parameters . . . 140
SMF Keyword Examples . . . 141
STREAM Keyword . . . 142
Syntax . . . 143
Keyword Name . . . 144
Parameters . . . 144
TMS Keyword . . . 155
Syntax . . . 155
Keyword Name . . . 155
Parameters . . . 155
Specifying a Time Interval or Period . . . 161
Syntax . . . 161
Keyword Name . . . 161
Parameters . . . 161
Examples . . . 163
File Example . . . 164

Chapter 7. Starting the ExHPDM Server . . . 175

Start Command . . . 175
Syntax . . . 176
Command Name . . . 176
Parameters . . . 176
START Command Examples . . . 179

Chapter 8. ExHPDM Operator Commands . . . 183

Entering ExHPDM Commands . . . 183
Command Syntax . . . 183

CANCEL Commands . . . 184
Syntax . . . 185
Command Name . . . 185
Parameters . . . 185
Cancel Example . . . 186
DISPLAY Commands . . . 186
Syntax . . . 187
Command Name . . . 187
Parameters . . . 187
DISPLAY Examples . . . 189
SET Command . . . 200
Syntax . . . 200
Command Name . . . 200
Parameters . . . 200
SET Examples . . . 203
SHUTDOWN Commands . . . 204
Syntax . . . 205
Command Name . . . 205
Parameters . . . 205
SHUTDOWN Command Examples . . . 205
START Command . . . 208
Syntax . . . 208
Command Name . . . 208
Parameters . . . 208

Chapter 9. ExHPDM Administration Utility (SOVADMN) . . . 211

Overview . . . 211
Running SOVADMN as a Batch Job . . . 211
Specifying an ExHPDM Server . . . 211
JCL Requirements . . . 212
Database Administration Commands . . . 216
DATABASE LIST, UPDATE, and DELETE Commands . . . 216
Syntax . . . 231
ADMIN DataBase UPDATE command . . . 234
ADMIN DATABASE Listing Options . . . 236
Other Administration Commands . . . 253
SCANSTReamfile and Disaster Recovery Mode . . . 254
Overview of ExHPDM Database Format and Operations . . . 265
Writing Client Files to the Database . . . 266
Reading Client Files from the Database . . . 267
Listing Objects in the Database . . . 267
Exceptional Conditions in the Database . . . 267
Considerations for Allocating the ExHPDM Database . . . 268
Reorganizing the ExHPDM Database . . . 271

Chapter 10. DD SUBSYS JCL Parameters . . . 273

Entering DD SUBSYS JCL 273
Syntax 274
Directive Name 275
Parameters 275
DD SUBSYS JCL Parameter Examples 280

Chapter 11. Changing MVS Parameter Files 283

Required Changes and Additions to MVS Files 283
MVS Subsystem Definition (IEFSSNxx) 283
Syntax 285
MVS Subsystem Definition 285
ExHPDM Subsystem Parameters 285
MVS Parameter File Examples 288
Example 1 288
Example 2 288
Example 4 289
Example 5 289
Example 7 290
Example 8 290

Appendix A: ExHPDM SMF Records 291

Outputs 291
SMF Mapping Macros 291
Common Data Areas 292
Audit Records 294
Accounting Records 296
Performance Records 298
Disaster Recovery Records 301

Index 1

Chapter 1. Introduction

Overview

Prior to ExHPDM, traditional backup and restore schemes used Operating Systems, third-party applications, and utilities that used logical data paths from files or disk storage to a single tape device. This scheme made the data path to the tape device dedicated to the application program for the duration of the backup or restore job. For example, if three storage volumes were backed up in parallel, three tape devices were needed. Also, the contents of the three storage volumes would be written to three separate tape files on three different tape devices.

This was a reasonable solution when the data transfer rate to tape devices was low in comparison to DASD. However, DASD data transfer rate is now matched or even exceeded by the new generations of modern tape devices.

Constraints within the MVS operating system have prevented files from being moved optimally between tape devices and DASD. This meant that the speed and capacity of tape devices have been greatly underutilized. ExHPDM makes what was previously impossible, possible because ExHPDM is a smarter generation of backup and restore software.

Unlike its predecessors in backup and restore software, ExHPDM is specifically designed to fully use the capabilities of modern tape devices, such as Sun/StorageTek's T10000s, T9940s, and T9840s. The increases in capacity and speed that ExHPDM delivers are exceptionally attractive benefits. For example, ExHPDM can drive Sun/StorageTek T10000 devices at a data transfer rate up to 200 MB per second, 9940 and 9840 devices up to 10 MB per second from DASD and arrays. Therefore, the time allocated for backup and restore operations can be lowered or more can be done within the existing time. Because storage media are being used to their optimum capacity, there is less overall media handling and management.

ExHPDM Description

ExHPDM is utility software that performs high-speed backup and restore of data sets by interleaving very large block sizes on high-speed, high-capacity tape devices. ExHPDM achieves its speed by treating all data equally regardless of the type. ExHPDM's only function is to move data from DASD to very fast tape and back again.

ExHPDM's version of the best method to move data is to enable tape devices to move data at their maximum available speed by:

- using 256 Kilobyte (KB) blocks or chunks of data
- interleaving the 256 KB blocks onto single or multiple tape volumes.

The ExHPDM software moves blocks of data in parallel from several concurrently executing MVS application programs. The data from the application programs is buffered into 256 KB tape block sizes in the application program's address space and the 256 KB blocks are interleaved onto single or multiple tape volumes.

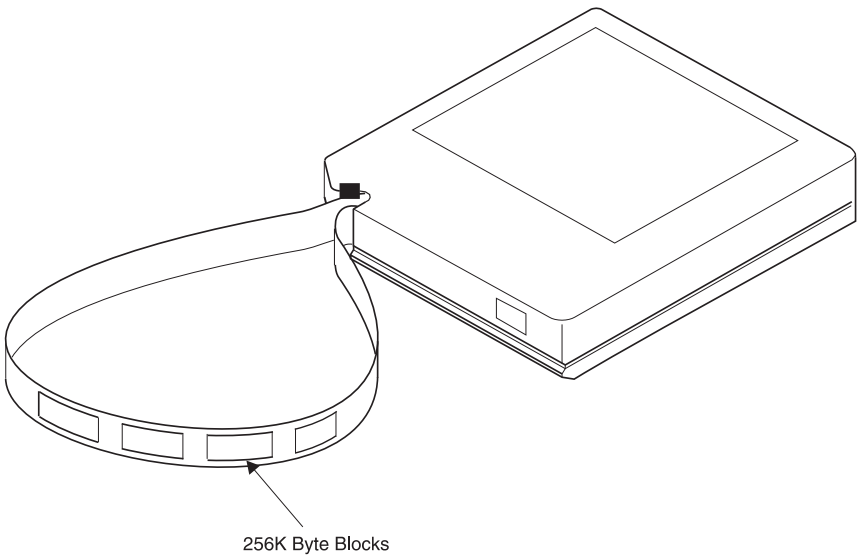


Figure 1. Interleaved 256 KB Blocks

Application program data is not intermixed in a block. Only the data from one application program is in a block. If the data from the application program does not fill a 256 KB block, the data is padded out to 256 KB. he use of the large block size exploits the 9840 and 9940 ESCON (13MB per second) interfaces and allows large-capacity tape volumes to be used effectively and efficiently. With 9840, 9940 FICON (100 MB per second) interfaces and indeed FICON/Express2 (2 GB per second) interfaces, a much greater exploitation of the channel bandwidth provides a greater level of performance with ExHPDM.

ExHPDM is optimized for high-speed backup operations and can be directed at both the IBM MVS DFSMSdss (ADRDSSU) product or the Innovation Data Processing Fast Dump and Restore (FDR) product.

DFSMSdss and FDR in ExHPDM

DFSMSdss and FDR have been used to stream data to tape, but they have only been able to do this by logical data paths. The use of ExHPDM and DFSMSdss or FDR dump and restore applications yields significantly reduced elapsed times for batch jobs compared

with traditional parallel schemes using multiple tape drives. ExHPDM does not replace the MVS DFSMSdss or FDR products, but works in conjunction with them and only for jobs where the use of ExHPDM is specified.

DFSMSdss and FDR use ExHPDM processing by a specification on a JCL DD statement using the SUBSYS parameter. DFSMSdss is indirectly invoked from an application program, called SOVDSSU, to cause redirection of I/O processing to ExHPDM. FDR is directly invoked without the need for an application program.

To DFSMSdss or FDR, ExHPDM appears to be an unlimited tape resource. This allows ExHPDM to simultaneously receive data from several different applications and place them onto the same tape volume on the same tape device.

ExHPDM Concepts

ExHPDM can be visualized as using a very wide-mouthed funnel, that is, the opening of a very large pipe where many application backup and restore jobs can be poured. The comparison of data movement to the way fluids flow is a very useful way of thinking about ExHPDM.

Understanding the following ExHPDM terms aids in understanding ExHPDM. These terms are:

- Administration Utility–SOVADMN
- DFSMSdss Interface–SOVDSSU
- Client
- Connection
- Expected Flow Increase
- Flow Limit
- Server
- Server Address Space
- Stream
- Stream Class
- Stream Device
- Stream File
- Stream Management
- Stream Task.

Administration Utility—SOVADMN

SOVADMN is a utility program that performs activities directly on the ExHPDM database and its tape media. SOVADMN is used as a batch application.

DFSMSdss Interface—SOVDSSU

Any batch job that uses IBM's DFSMSdss DASD to tape data backup utility, ADRDSSU, can take advantage of ExHPDM. A standard DFSMSdss job that performs backup and restores can be used. This use is done by performing a connection at the JCL DDNAME level to the ExHPDM server.

A special ExHPDM DFSMSdss interface routine, called a DFSMSdss UIM (User Interface Module), is named SOVDSSU. To use ExHPDM, JCL that requires DFSMSdss should be changed to use the program name of SOVDSSU. Then, the full benefits of ExHPDM block interleaving are realized. SOVDSSU accepts all of the standard DFSMSdss directives.

ExHPDM Client

An ExHPDM client is a DFSMSdss or FDR job in MVS that requests data set processing by ExHPDM. The client is typically an MVS batch job. Client requests can be either for reading from a tape volume or writing to a tape volume.

A client can have one or more connections to one or more ExHPDM servers. Furthermore, client connections are made through the use of the JCL DD SUBSYS keyword whose parameters are processed by ExHPDM. ExHPDM uses the information presented on the JCL DD SUBSYS keyword and parameters in its startup parameter file to determine how to process the client connection.

Clients are synchronized by the server so a request appears to be to or from an actual tape device. A client application program performs allocation, open, close, and deallocation for an ExHPDM processed file causing the ExHPDM server to establish a connection between the client's data management access method and itself.

The client performs the access method I/O when logical or physical record (blocks) requests are processed synchronously and asynchronously by the ExHPDM server.

The functions of any ExHPDM client are to:

- connect and disconnect to and from the ExHPDM server
- pass any parameters specified on the SUBSYS parm
- perform access method I/O
- propagate data management status to the application

Note: ExHPDM client processing messages are returned in the clients JESYSMSG data set and, by default, to the ExHPDM server logfile.

ExHPDM Client Write Processing

Clients that have their program output data sets redirected to ExHPDM processing have their logical records or physical blocks processed by the ExHPDM server. The access method used by the client has its data records buffered for processing in its address space. The operation is transparent and is synchronous to the client.

ExHPDM Client Communications

Communication is initiated by the client to the ExHPDM server through the specification of ExHPDM data management parameters on the data set JCL specification using the MVS subsystem data set facility. These parameters are specified on the DD SUBSYS statement in MVS JCL.

Connection

A connection is like a pipe constructed between a client's JCL DD statement and the ExHPDM server. This connection is made when the job is made known to a particular ExHPDM server because of the use of the JCL DD SUBSYS keyword and its parameters.

Expected Flow Increase

Expected Flow Increase (EFI) is the amount of data per second that a connection is likely to be. EFI allows load balancing to be performed by ExHPDM to optimize the usage of output tape devices. For example, consider a stream task with three simultaneous connections, each of which have an average data rate of 1.5 MB per second from the DASD subsystem. In ExHPDM's terms, each JCL DD statement from a client that connects to a particular stream increases the stream's overall data rate by 1.5 MB per second. The user specifies to ExHPDM via the CLASS EFI keyword that the connections for that stream average 1.5 MB per second; for example: CLASS(CLASS1 EFI(1.5M)). For the three connections the accumulated EFI to the stream is 4.5 MB per second.

If enough connections are added such that the Expected Flow Increase is greater than the amount of data that a stream device can sustain (Flow Limit), ExHPDM either rejects further connections to the stream, puts the connections in a wait status, or schedules more tape devices to spread the load. No matter what action is taken, ExHPDM's action is dependent on the specified Expected Flow Increase.

Flow Limit

Flow Limit is the amount of data that the stream device can sustain. For example, a Timberline 9490 could have a Flow Limit specified at up to 7MB per second, a 4490 tape device could have its Flow Limit specified as 3MB per second, a 9840 and 9940 (ESCON) could have a Flow Limit specified at up to 10 MB per second, 9840 and 9940 (FICON) could have a Flow Limit specified at up to 50MB per second, and a T10000 (FICON/Express2) could have a Flow Limit specified at up to 200MB per second.

Note: ExHPDM still supports Redwood devices. In the above example, if a Redwood SD–3 helical tape device is available for a stream, its Flow Limit could be specified at 12 MB per second.

ExHPDM MVS Server

The ExHPDM server takes advantage of high-speed tape devices by writing interleaved 256 KB blocks from multiple ExHPDM client connections onto multiple tape volumes. The server processes multiple streams (reading from a tape volume and writing to a tape volume) using criteria specified by the operator or by the ExHPDM startup parameter file to:

- schedule
- consolidate types of data per stream
- regulate performance.

The ExHPDM server is implemented as an MVS address space. The server may be started as follows:

- either on demand by the MVS operator or automated operator software
- or through the MVS subsystem specification when MVS is initialized through the Master Subsystem Initialization (MSI) as an MVS full function address space.

The ExHPDM server physically reads and writes client requests directly from the client address space to a tape device. Client write requests are buffered in the client's address space until a large 256 KB block is ready to be scheduled for writing to an actual tape device.

For read requests, the ExHPDM server retrieves the specified 256 KB block from tape and makes it available to the application program. The ExHPDM client access method support extensions return the client's data to the application program as either logical or physical records as required by the access method used in the client.

The ExHPDM server maintains information in a VSAM database that can be shared between other ExHPDM servers on the same or different MVS images. This database contains an inventory of all files and data sets that are managed by ExHPDM. The database is also used to maintain the tape volumes used to contain all the data managed by ExHPDM. The ExHPDM server must have a database.

ExHPDM Server Communications

The ExHPDM server provides its service to ExHPDM clients through a ExHPDM API known internally as VSTAM.

ExHPDM Subsystem API

The ExHPDM server uses the MVS subsystem interface. The server provides MVS data management and operator command extensions through the MVS subsystem interface.

Stream

A Stream is a logical grouping of client connections for the purposes of providing data segregation and load balancing while writing to a tape device. Using streams allows specific data associated with a JCL DD statement to go to an actual tape device used by the ExHPDM server. For example, a particular MVS system has a range of data types and backup needs that are:

- Full weekly system volume–high speed
- Production video data–high volume
- Small daily incremental files used by programmers–low speed and low volume.

The full weekly system volume backups can be directed to a 9840 drive.

The production video data is contained over a large DASD array and could be about 150 to 300 gigabytes (GB). This backup could be directed to the 9840 and 9940A drives.

The small daily incremental files used by programmers can all be backed up by low capacity and low speed tape devices.

When ExHPDM is customized and the job entered into the MVS system for processing by ExHPDM, ExHPDM starts all of these different streams automatically. Any number of streams may be active at the same time. An active stream is called a stream task. There can only be one stream task assigned to an actual tape device. If the stream type is unknown, you are able to either have a default stream process the connection or reject the connection outright. When connection to a stream is rejected, the client is notified in the job's JESYSMSG data set.

Stream Class

Stream Class defines the performance (Expected Flow Increase) and management characteristics of a connection and the stream to connect to.

Stream Device

A Stream Device is the tape device specification to be used for a particular stream. This is usually specified as an esoteric name for a tape device known to the user. This is where the tape device's characteristics (such as speed, data rate, and capacity) impact upon the specified stream class and task allocation.

Stream File

A Stream File is a collection of tape volumes written by a stream task.

Stream Management

Stream Management defines the management attributes for a client data set.

Stream Task

A Stream Task is the unit of work that processes an instance of a stream. The stream task controls the writing and reading of a specific set of tapes used by a stream.

ExHPDM Database

The ExHPDM server database is a VSAM KSDS that can be shared among multiple ExHPDM servers on multiple MVS images. The ExHPDM database should be defined after ExHPDM is installed and before ExHPDM is started. The database may be defined using the sample SOVDBDEF in the STKSAMP data set.

The ExHPDM server should have a database. ExHPDM may be started, but does not have full functionality without its database unless in disaster recovery (DR) mode. Refer to the

Installation Manual and “Considerations for Allocating the ExHPDM Database” on page 268 for more details.

ExHPDM Journal Data Set

The ExHPDM journal data set is a sequential data set that can be individually defined for a particular ExHPDM server. When multiple ExHPDM servers are operating, a journal data set should be defined for each server if journaling is specified. While it is optional to use a journal, one should be used in case of database corruption or a media problem.

Using Non–ExHPDM Utilities for the ExHPDM Database

Warning: You **MUST NOT** use MVS VSAM utilities to backup and restore the ExHPDM database. The use of non ExHPDM utilities corrupts the internal structure used by ExHPDM to efficiently use the space and evenly distribute the ExHPDM database records.

Warning: Utilities, such as MVS DFSMShsm, DFSMS/MVS Access Method Services (IDCAMS) and ISV VSAM enhancement, optimization, and replacement products must not be used.

ExHPDM Startup Parameter File

The ExHPDM server uses a parameter file on startup to define the way that it should be run. This parameter file is stored and used on the MVS system(s) where ExHPDM is started. The startup parameter file may be stored in the MVS parameter library SYS1.PARMLIB or a user-defined library, as specified in the ExHPDM server startup procedure. Refer to “Chapter 6. ExHPDM Startup Parameter File” on page 89 for details on defining the parameter file.

ExHPDM and Z/OS 1.9

Z/OS 1.9 provides the REFRPROT statement type to specify that REFR programs are protected from modification by placing them in key 0, non-fetch protected storage, and page protecting the full pages. **Note that** the REFRPROT statement causes ExHPDM to fail, so ensure that you specify the NOREFRPROT parameter for ExHPDM.

Chapter 2. License Key

ExHPDM License Key

Upon installation, you have a 75-day trial period. During this 75-day period, you must secure a permanent license key. Do not wait until this trial period is over to secure your permanent key. Once this trial period expires you cannot start ExHPDM.

The two types of license keys are:

- Permanent: Enables you to use ExHPDM when StorageTek has received the initial license fee for a product or feature.
- Emergency Software Key: For short term use in emergency situations. It is limited to seven days.

Note: After the trial period expires and a valid license key has not been entered or an emergency license key expires, ExHPDM automatically shuts down. When the permanent key expires, no new output connections are allowed. However, you are still able to perform client restores and administrative functions.

Serial Number

Encoded in the license key is a serial number unique to you and the product. It can be up to 12-digits in length. This number must be quoted to obtain software support. The serial number is displayed, along with other license key information when ExHPDM validates the license key information. This is done at startup or if the license key information is changed though a SET PRM command. It is also displayed as part of the output produced from the DISPLAY VERSION command.

Obtaining a Permanent License Key

1. Access the StorageTek Customer Resource Center (CRC) at the following URL http://www.support.storagetek.com.
2. Click Tools & Services.
3. Click Software Keys.
4. Scroll to ExHPDM and click ExHPDM *x.x.x* Key Request.

   The ExHPDM *x.x.x* Key Order Form displays.

5. Click Permanent.
6. Complete the following mandatory information. Note: ExHPDM is started without this information only if you are within the trial period.
    - LKEYINFO (PRODUCT(*product _name*)

      This is the ExHPDM version and release numbers.
    - CUSTOMER (customer_site_name)

      This is your 20-character customer name. If the name contains blanks, you must enclose it in single quotes.
    - SITE (*site_numbe*r)

      This is the site number assigned to you by SMD for your license key and product. The site number consists of four to six numeric characters.
    - EXPIRY (*expiry_date*)

      This is the expiration date in ISO standard YYYYDDD format. Example: 2006009
    - KEY (*license_key*)

      This is your key that is assigned by SMD and is up to 24-characters in length. Keys are issued within 48 hours of the license key request, Monday through Friday, 7:00 a.m. to 4:00 p.m. Mountain Time, except holidays.

Obtaining an Emergency License Key

1. Access the StorageTek Customer Resource Center (CRC) at the following URL http://www.support.storagetek.com.
2. Click Tools & Services.
3. Click Emergency Software Key.

   The StorageTek Emergency Generator screen displays

   All fields with an asterisk are required and must be completed before submitted the license key request.

License Key Validation

ExHPDM validates the license key upon entry, during product initialization, and when the license key information is changed due to a SET PRM command. It checks the expiry date of the key at midnight each day and when it validates the key. The product will not initialize at system startup if you do not have the appropriate product/feature license key. While the ExHPDM system is up and running, warning messages are issued as the license key expiry date approaches. You will need to update your license key.

If you experience errors with your ExHPDM license key:

- Consult the *ExHPDM Messages Manual,* and follow the instructions for the error message you received.
- If you are still unable to resolve the problem, contact the StorageTek Software Manufacturing and Distribution department (1-800-436-5554) and select option three, or call your StorageTek Marketing Representative or Systems Engineer during normal business hours.

Chapter 3. Operation

Overview

ExHPDM provides the streaming of interleaved blocks of data from a number of clients through server stream tasks to real output devices. Each block only contains the data from one client. The basic structure and components of ExHPDM are shown in Figure 2.

Figure 2 shows many DFSMSdss and clients running concurrently using the stream interleaving facility of ExHPDM. This implementation fully uses the performance benefits of high performance channels and tape devices (such as Sun/STK T10000 (FICON/Express2), 9940 and 9840 (FICON) devices).

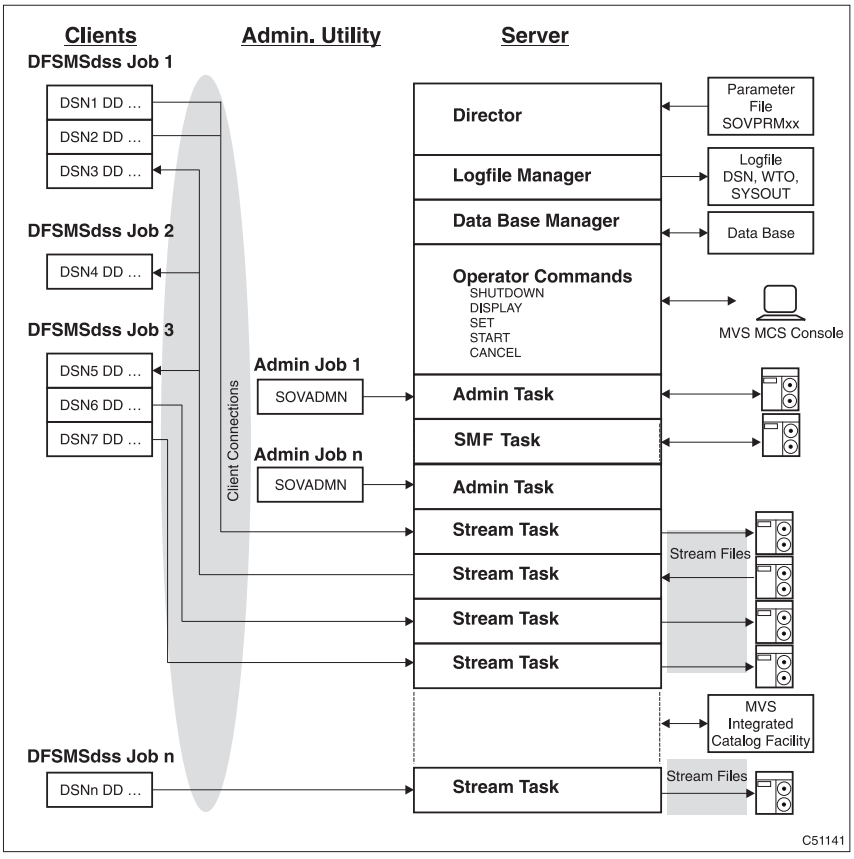


Figure 2. Basic Structure and Components of ExHPDM

ExHPDM Processing

ExHPDM automatically selects the appropriate stream task to process a write request and places the data into a ExHPDM managed file called the stream file. The stream file name is selected by ExHPDM based on the stream task parameters. In addition, parameters can be specified on the clients DD SUBSYS statement to influence the selection and processing options used by the ExHPDM server.

When a client closes a data set that was written to a stream file, ExHPDM conditionally catalogs the client's new data set in the MVS catalog, as specified in the JCL disposition. If a client data set disposition is specified as DISP=(,KEEP,*xx*) or DISP=(,CATLG,*xx*), the client's data set is always cataloged in the Integrated Catalog Facility (ICF) catalog. The client's data set is cataloged using the dummy volume serial (VOLSER) **SOV*.

All client data sets should be cataloged. Cataloging client data sets is required when using Generation Data Group (GDG) data sets because the ICF catalog identifies the versions of the client's data set.

The client data set information is always maintained in the ExHPDM database. This database consists of both client and stream information. Client information includes the position (block ID) within the output stream file. Stream information includes media type and volume serial. The client data set information maintained in the ExHPDM database is used to satisfy a client read request.

When ExHPDM receives a client connection request to read from a tape volume, it obtains the client data set details from its database to locate the stream file name, the associated volumes, and the starting block ID (for fast locate). ExHPDM automatically mounts and reads the blocks from the stream file and passes them to the client connection to be processed. When the client completes its connection (close), ExHPDM performs resource cleanup and, if no other clients are also using the same stream task, the stream file is closed.

ExHPDM consists of three major components:

- ExHPDM Clients
- ExHPDM Administration Utility
- ExHPDM Server.

ExHPDM Clients

An ExHPDM client is any address space in MVS (STC, BATCH, LOGON, or full function SASI) that contains at least one connection to ExHPDM.

A client is composed of one or more connections (DD names) that are to be processed by ExHPDM. A client can have many connections to one or more ExHPDM servers using different DD SUBSYS parms.

One or more client DD names are connected to ExHPDM through processing at the DFP access method level using DD SUBSYS processing. A logical connection is established to the ExHPDM server for input (read) or output (write) processing.

This release of ExHPDM contains support for SMC V6.2 to set dynamically via policies, the DD SUBSYS specification to direct the processing to ExHPDM. This new feature provides customers with the ability of running client jobs, without having to alter their existing backup/restore jobs to make use of ExHPDM. For further information on these SMC policy directives refer to the SMC 6.2 Configuration and Administration Guide.

DFSMSdss

ExHPDM clients are only processed for ADRDSSU (DFSMSdss) requests through the use of the SOVDSSU program provided by ExHPDM. ExHPDM supports connections from DFSMSdss for the following directives:

- DUMP
- RESTORE
- COPYDUMP.

Any DFSMSdss directives other than these can still be processed using SOVDSSU; however, no connections for these are made to ExHPDM. ExHPDM supports write (OUTDDNAME) connections for DUMP, read (INDDNAME) connections for RESTORE, and read and/or write connections for COPYDUMP.

Where COPYDUMP input is from a ExHPDM client file, the COPYDUMP output must be contained on a device that can handle the size of blocks from the input. This means that in general, a DFSMSdss DUMP in an ExHPDM stream file cannot be COPYDUMP'd to a DASD volume (i.e., the block size used by DFSMSdss when interfacing with ExHPDM is generally 65520 bytes). The DFSMSdss error message returned in such a circumstance is ADR331E. For example:

ADR331E (002) -CPYD (01). OUTPUT BLKSIZE 47968 FOR DATA SET ON DDNAME SUBS1 SMALLER THAN INPUT BLKSIZE 65520

Note:

- When using COPYDUMP, the recommendation is to use WAIT(NO) on the DD SUBSYS connection parameters or on the STREAM definitions when ExHPDM is used for both the INDDNAME and OUTDDNAME. This is to prevent delayed connections from hanging up or causing a deadlock condition with other processing. If WAIT(YES) is specified, it is possible to get a deadlock condition when all connections cannot be satisfied.
- When constructing the COPYDUMP jobs, its important to note whether SEPARATEBYJOB is used or not when the DUMPS were produced. When SEPARATEBYJOB has not been used to create the ExHPDM STREAM, the client dumps can be spread across many streams. You can improve the performance of the COPYDUMPs by creating the batch jobs where the input datasets containing all of the client dumps are on the same stream. You should also pay attention to the DEVICELIMIT specification for this workload. Too small a number can result in a deadlock situation when WAIT(NO) is specified, when the device limit has been reached. When this occurs, those clients will be terminated, and can appear to cause a deadlock situation.

The selection rules in the ExHPDM server startup parameter file are used to include or exclude DFSMSdss output data sets for ExHPDM processing. DFSMSdss generates output data when using the DUMP and COPYDUMP OUTDDNAME references.

Included output data sets become client connections to an ExHPDM stream task. Excluded data sets are not processed by ExHPDM. See “Chapter 6. ExHPDM Startup Parameter File” on page 89 for a description of the startup parameter file.

No selection rules are required for ExHPDM managed DFSMSdss INDDNAME data sets as ExHPDM already knows how to process the data set due to the original write processing.

User Interface Module (UIM)

The use of a special DFSMSdss interface routine, enables ExHPDM to process write requests from DFSMSdss dumps and read requests to restore the data. This routine is called a DFSMSdss UIM (User Interface Module).

The DFSMSdss UIM is a standard documented interface that ExHPDM uses to initialize and redirect normally physically written and read data blocks to or from the ExHPDM server stream tasks. The selection of the DFSMSdss UIM connections to process with ExHPDM can be done through the use of the DD SUBSYS JCL directive and/or through the use of the ExHPDM selection rules.

The ExHPDM DFSMSdss interface is contained with the module named SOVDSSU. To use this interface, JCL that would normally specify DFSMSdss (ADRDSSU) should be changed to specify the ExHPDM program name of SOVDSSU. By using the SOVDSSU program, the full benefits of ExHPDM block interleaving is realized.

The ExHPDM DFSMSdss interface can be used to backup and restore HFS data. Additionally, at z/OS 1.3 and above the ExHPDM DFSMSdss interface can backup and restore Linux volumes or Linux partitions under the following environments.

- A z/OS-centric environment with z/OS running on several LPARs, in tandem with a few dozen Linux servers running within the Virtual Image Facility (VIF).
- A Linux-focused environment with VM running in BASIC or LPAR mode. Hundreds of Linux guests and one or more z/OS images will perform the DFSMSdss processing.

Examples

Example 1

The following full volume dump example uses the SOVDSSU DFSMSdss interface. The connection to ExHPDM is requested through the **DD SUBSYS=(SOV)** keyword:

```
//DUMPERS  EXEC PGM=SOVDSSU,REGION=5M
//SYSPRINT DD SYSOUT=*
//SUBSYSDD DD SUBSYS=(SOV),DSN=MVSRES.DUMP,
//          DISP=(,CATLG)
//SYSIN    DD *
 DUMP FULL INDYNAM(MVSRES)    -
```

```
    OUTDDNAME( SUBSYSDD)  -
    OPTIMIZE(1) WAIT(2,2)
/*
```

The following ExHPDM parameters may be specified in the startup parameter file for the above example. The selection rule (RULE1) can be used to relate the **MVSRES.DUMP** data set to a particular ExHPDM stream class (CLASS1); this is, in turn, connected to an ExHPDM stream (STRM1) through the class definition.

```
SELECT(RULE1  DSN(MVSRES.DUMP) CLASS(CLASS1))
CLASS(CLASS1 STREAM(STRM1))
```

Example 2

The following full volume restore example uses the dump data set MVSRES.DUMP, as created by the previous example, to restore the MVSRES volume. ExHPDM does not need to use any SELECT rules or stream CLASS to process the data set for input processing. All the information required by ExHPDM to processes the data set is contained in its database. The connection to ExHPDM is requested through the DD SUBSYS=SOV keyword:

```
//RESTORE EXEC PGM=SOVDSSU,REGION=5M
//SYSPRINT DD SYSOUT=*
//SUBSYSDD DD SUBSYS=SOV,DSN=MVSRES.DUMP,
//          DISP=SHR
//SYSIN    DD *
 RESTORE INDDNAME(SUBSYSDD) OUTDYNAM(MVSRES)
/*
```

Example 3

The following physical data set dump example results in the output data written to **SUBSYSDD** to be handled by an ExHPDM server stream task

```
//DUMPERS  EXEC PGM=SOVDSSU,REGION=5M
//SYSPRINT DD SYSOUT=*
//SUBSYSDD DD SUBSYS=(SOV),DSN=OWPL.DUMP(+1),
//        DISP=(,CATLG)
//SYSIN    DD *
 DUMP DATASET( INCLUDE( PLJRA.PRIVATE.**))  -
        OUTDDNAME( SUBSYSDD )  -
         CANCELERROR OPTIMIZE(1) WAIT(2,2)
```

The following ExHPDM parameters may be specified in the startup parameter file for the above example. The selection rule (DSSRULE) can be used to relate the **OWPL.DUMP** GDG data set to a particular ExHPDM stream class (DSSCLASS); this is, in turn, connected to an ExHPDM stream (DSSSTRM) through the class definition. Masking is used for the SELECT DSN to allow for the GDG G*xxxx*V00 appended to OWPL.DUMP.

```
SELECT(DSSRULE  DSN(OWPL.DUMP.**) CLASS(DSSCLASS))
CLASS(DSSCLASS STREAM(DSSSTRM))
```

The syntax of the SELECT keyword is described in “SELECT Keyword” on page 134. The syntax of the CLASS keyword is described in “CLASS Keyword” on page 99.

Example 4

The following copy dump example results in the dump from Example 3, as written to **SUBSYSDD,** to be copied to **SUBSYSCP** by an ExHPDM server stream task. Note the use of WAIT(NO) on the INDDNAME and OUTDDNAME. This prevents other processing from being held up or a deadlock occurring when stream file resources for these connections cannot be obtained.

```
//DUMPCOPY EXEC PGM=SOVDSSU,REGION=5M
//SYSPRINT DD SYSOUT=*
//SUBSYSDD DD SUBSYS=(SOV,'WAIT(NO)'),DSN=OWPL.DUMP(0)
//SUBSYSCP DD SUBSYS=(SOV,'WAIT(NO)'),DSN=OWPL.DUMP.COPY(+1),
//        DISP=(,CATLG)
//SYSIN    DD *
 COPYDUMP -
        INDDNAME( SUBSYSDD )-
         OUTDDNAME( SUBSYSCP )
```

The same ExHPDM selection rules, stream class and stream may be used as in Example 3.

Exploiting Parallelism with ExHPDM

ExHPDM works most efficiently with DFSMSdss when directives are processed in parallel ("*simultaneously*") instead of serially. This may be done with multiple jobs or by using the DFSMSdss PARALLEL directive. This directive is available and required for the most efficient use of ExHPDM.

DFSMSdss might change PARALLEL processing to SERIAL processing when it detects that similar resources are being used by multiple DFSMSdss directives in the same job step. Typically, this occurs if the same volume serial is referenced by more than one DUMP or RESTORE directive. DFSMSdss assumes there will be a resource conflict and will make the processing occur serially.

DFSMSdss changes DUMP processing from PARALLEL to SERIAL, when the same volume(s) are specified in different DUMP directives by one of the following DFSMSdss keywords:

- INDDNAME
- INDYNAM
- LOGINDDNAME
- LOGINDYNAM.

DFSMSdss changes RESTORE processing from PARALLEL to SERIAL, when the same volume(s) are specified in different RESTORE directives by one of the following DFSMSdss keywords:

- OUTDDNAME
- OUTDYNAM.

If the same volume(s) are to be specified in multiple DUMP or RESTORE directives, then multiple jobs may be used to make the DFSMSdss processing execute in parallel.

For example, the following DFSMSdss DUMPs run serially even though PARALLEL is specified. This occurs because the same LOGINDYNAM volume, VOL003, is specified by the two DUMP directives. DFSMSdss assumes there will be a resource conflict, so it turns these into a serial process. To make these dumps run in parallel, either the VOL003 volume would need to be removed from one of the DUMP directives or multiple jobs would need to be run.

```
PARALLEL
DUMP DATASET(INCLUDE(APP1.DS*.**)) -
   LOGINDYNAM((VOL001),(VOL002),(VOL003))
DUMP DATASET(INCLUDE(APP2.DS*.**)) -
   LOGINDYNAM((VOL003),(VOL004),(VOL005))
```

Examples

The following rule applies to the examples to relate the ****.RD3A.**** data set in DFSMSdss ExHPDM processing to a particular stream class; this is, in turn, connected to ExHPDM through a class definition.

```
SELECT(DSSRULE  DSN(**.RD3A.**) CLASS(DFDSS))
CLASS(DFDSS STREAM(DFDSS))
STREAM(DFDSS DSN(STREAM.DS) CONCURRENT(WRITE(1)))
```

The syntax of the SELECT keyword is described in “SELECT Keyword” on page 134. The syntax of the CLASS keyword is described in “CLASS Keyword” on page 99. The syntax of the STREAM keyword is described in “STREAM Keyword” on page 142.

Example 1

In the following example, three full volume dumps are processed in parallel by running three DFSMSdss jobs. Their output is directed to the same stream task in ExHPDM.

```
//DUMPJOB1  JOB jobcard info
//* Job backs up 1 pack in parallel
//* backups will be to a single stream file due to
//*          concurrent(write(1))
//STEP001  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//DD01     DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//OUT01    DD  SUBSYS=(SOV),DSN=DUMP.RD3A.FULL.HPDM01,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 DUMP FULL INDD(DD01) OUTDD(OUT01)
/*
//DUMPJOB2  JOB jobcard info
//* Job backs up 1 pack in parallel
//* backups will be to a single stream file due to
//*                   concurrent(write(1))
//STEP002  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//DD02     DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
```

```
//OUT02    DD  SUBSYS=(SOV),DSN=DUMP.RD3A.FULL.HPDM02,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 DUMP FULL INDD(DD02) OUTDD(OUT02)
/*
//DUMPJOB3  JOB jobcard info
//* Job backs up 1 pack in parallel
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP003  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//DD03     DD  UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//OUT03    DD  SUBSYS=(SOV),DSN=DUMP.RD3A.FULL.HPDM03,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 DUMP FULL INDD(DD03) OUTDD(OUT03)
/*
```

Example 2

In the following example, three full volume dumps from the previous example are processed in parallel by DFSMSdss using the PARALLEL directive and one job. Their output is directed to the same stream task in ExHPDM.

```
//DUMPJOB  JOB jobcard info
//* Job backs up 3 packs (in parallel)
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP001  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//DD01     DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DD02     DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DD03     DD  UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//OUT01    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//         DISP=(,CATLG,DELETE)
//OUT02    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
//         DISP=(,CATLG,DELETE)
//OUT03    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM03,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 PARALLEL
 DUMP FULL INDD(DD01) OUTDD(OUT01)
 DUMP FULL INDD(DD02) OUTDD(OUT02)
 DUMP FULL INDD(DD03) OUTDD(OUT03)
/*
```

Example 3

In the following example, four logical volume dumps are processed in parallel by running three DFSMSdss jobs. Three jobs are required as the LOGINDYNAM volumes for application dumps for LDMPJOB2 (PL.APP3) and LDMPJOB3 (PL.APP4) are also referenced in the application dumps in LDMPJOB1 (PL.APP1 and PL.APP2). If these dumps were to be in a single dump job, then DFSMSdss assumes that a resource conflict is going occur and would force them to be done serially even when PARALLEL is specified. Their output is directed to the same stream task in ExHPDM.

```
//LDMPJOB1  JOB jobcard info
//* Job backs up logical data sets for applications APP1 and APP2
//* in parallel with dump jobs LDMPJOB2 and LDMPJOB3.
//* Backups will be to a single stream file due to
//*                 concurrent(write(1))
//STEP001  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//OUT01    DD  SUBSYS=SOV,DSN=DUMP.RD3A.LOGICAL.APP1,
//         DISP=(,CATLG,DELETE)
//OUT02    DD  SUBSYS=SOV,DSN=DUMP.RD3A.LOGICAL.APP2,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 PARALLEL
 DUMP DATASET(INCLUDE(PL.APP1.**)) -
        LOGINDYNAM((HPDM01),(HPDM02)) OUTDD(OUT01)
 DUMP DATASET(INCLUDE(PL.APP2.**)) -
 LOGINDYNAM((HPDM03),(HPDM04)) OUTDD(OUT02)
/*
//LDMPJOB2  JOB jobcard info
//* Job backs up logical data sets for application APP3
//* in parallel with dump jobs LDMPJOB1 and LDMPJOB3.
//* Backups will be to a single stream file due to
//*                 concurrent(write(1))
//STEP002  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//OUT03    DD  SUBSYS=SOV,DSN=DUMP.RD3A.LOGICAL.APP3,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 DUMP DATASET(INCLUDE(PL.APP3.**)) -
        LOGINDYNAM((HPDM01)) OUTDD(OUT03)
/*
//LDMPJOB3  JOB jobcard info
//* Job backs up logical data sets for application APP4
//* in parallel with dump jobs LDMPJOB1 and LDMPJOB2.
//* Backups will be to a single stream file due to
//*                 concurrent(write(1))
//STEP003  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//OUT04    DD  SUBSYS=SOV,DSN=DUMP.RD3A.LOGICAL.APP4,
//         DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 DUMP DATASET(INCLUDE(PL.APP4.**)) -
        LOGINDYNAM((HPDM03)) OUTDD(OUT04)
/*
```

Example 4

HFS Backup

```
//LDMPJOB4 JOB jobcard info
//* Job backs up HFS data
//STEP1     EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT  DD  SYSOUT=*
//IN1       DD  DSN=HFS.MVS.ROOT,DISP=SHR
//TAPE1     DD  DSN=DUMP.HFS.DATA,DISP=(,CATLG,DELETE),
```

```
//                UNIT=TAPE,SUBSYS=SOV
//SYSIN     DD *
 DUMP DATASET(INCLUDE(**)) INDD(IN1) OUTDD(TAPE1) ALLEXCP ALLDATA(*)
/*
```

Example 5

Linux Backup

```
//LDMPJOB4 JOB jobcard info
//* Job backs up Linux partition
//STEP1    EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//IN1      DD  UNIT=3390,VOL=SER=LINUX1,DISP=SHR
//TAPE1    DD  DSN=DUMP.PART.DATA,DISP=(,CATLG,DELETE),
//         SUBSYS=SOV
//SYSIN    DD *
DUMP DATASET(INCLUDE(LINUX.VLINUX1.PART0000.NATIVE))INDD(IN1)
        OUTDD(TAPE1) ALLEXCP
/*
```

FDR

Connections are made to ExHPDM through FDR by the DD SUBSYS (MVS subsystem) interface. FDR, unlike DFSMSdss with SOVDSSU, requires no special ExHPDM interface program.

FDR support of ExHPDM is integrated into FDR V5.3 L31, and above, to support the following FDR functions:

- FDR
- FDRDSF
- FDRTCOPY
- FDRCRYPT

FDRABR support of ExHPDM is integrated into FDR V5.3 L40, and above, to support the following FDRABR functions:

- FDRABR
- FDRAPPL
- FDRTSEL
- FDRTCOPY

Note: A license for FDR InstantBackup is required to use ExHPDM with FDRABR.

The selection rules in the ExHPDM server startup parameter file are used to include or exclude FDR output data sets from being considered for ExHPDM processing. FDR generates output data for data sets defined for the following DDs:

- TAPE*x* and TAPE*xx* (for duplexing), when performing DUMP processing,

```
SELECT(RULE1  DSN(MVSRES.DUMP) CLASS(CLASS1))
CLASS(CLASS1 STREAM(STRM1))
```

Example 2

The following full volume restore example uses the dump data set MVSRES.DUMP, as created by the previous example, to restore the MVSRES volume. ExHPDM does not need to use any SELECT rules or stream CLASS to process the data set for input processing. All the information required by ExHPDM to processes the data set is contained in its database. The connection to ExHPDM is requested through the DD SUBSYS=SOV keyword:

```
//RESTORE    EXEC PGM=FDR,REGION=5M
//SYSPRINT   DD SYSOUT=*
//DISK1      DD UNIT=SYSDA,DISP=OLD,VOL=SER=MVSRES
//TAPE1      DD SUBSYS=SOV,DSN=MVSRES.DUMP,
//             DISP=SHR
//SYSPRIN1   DD SYSOUT=*
//SYSIN      DD *
 RESTORE     TYPE=FDR,MAXERR=1
/*
```

Example 3

The following example shows the usage of FDRDSF. The specification of DD SUBSYS on **TAPE1** indicates that output should be handled by an ExHPDM server stream task.

```
//DUMPDSF    EXEC PGM=FDRDSF,REGION=5M
//SYSPRINT   DD SYSOUT=*
//DISK1      DD UNIT=SYSDA,DISP=OLD,VOL=SER=TSTVOL
//TAPE1      DD SUBSYS=(SOV),DSN=OWPL.DUMP(+1),
//             DISP=(,CATLG)
//SYSPRIN1   DD SYSOUT=*
//SYSIN      DD *
 DUMP        TYPE=DSF,DSNENQ=USE,MAXERR=1
 SELECT      DSN=JRA.**
/*
```

The following ExHPDM parameters may be specified in the startup parameter file for the above example.The selection rule (DSFRULE) can be used to relate the **OWPL.DUMP** GDG data set to a particular ExHPDM stream class (FDR); this is in turn connected to an ExHPDM stream (FDR) through the class definition.

```
SELECT(DSFRULE  PGM(FDRDSF) CLASS(FDR))
CLASS(FDR STREAM(FDR))
```

Note the usage of the PGM keyword in the select rule. This part of the example may be used when using multiple programs.

data set to a particular ExHPDM stream class (CLASS1); this is in turn connected to an ExHPDM stream (STRM1) through the class definition.

```
SELECT(RULE1  DSN(MVSRES.DUMP) CLASS(CLASS1))
CLASS(CLASS1 STREAM(STRM1))
```

The syntax of the SELECT keyword is described in . The syntax of the CLASS keyword is described in .

Example 2

The following full volume restore example uses the dump data set MVSRES.DUMP, as created by the previous example, to restore the MVSRES volume. ExHPDM does not need to use any SELECT rules or stream CLASS to process the data set for input /*

Example 4

The following example shows the usage of FDRDSF. The specification of DD SUBSYS on **TAPE1** indicates that output should be handled by an ExHPDM server stream task.

```
//DUMPDSF  EXEC PGM=FDRDSF,REGION=5M
//SYSPRINT DD SYSOUT=*
//DISK1    DD UNIT=SYSDA,DISP=OLD,VOL=SER=TSTVOL
//TAPE1    DD SUBSYS=(SOV),DSN=OWPL.DUMP(+1),
//            DISP=(,CATLG)
//SYSPRIN1 DD SYSOUT=*
//SYSIN    DD *
 DUMP      TYPE=DSF,DSNENQ=USE,MAXERR=1
 SELECT    DSN=JRA.**
/*
```

The following ExHPDM parameters may be specified in the startup parameter file for the above example.The selection rule (DSFRULE) can be used to relate the **OWPL.DUMP** GDG data set to a particular ExHPDM stream class (FDR); this is in turn connected to an ExHPDM stream (FDR) through the class definition.

```
SELECT(DSFRULE  PGM(FDRDSF) CLASS(FDR))
CLASS(FDR STREAM(FDR))
```

Note the usage of the PGM keyword in the select rule. This part of the example may be used when using multiple programs.

Example 5

The following example shows the usage of FDRTCOPY to copy a previously streamed FDR dump. The specification of DD SUBSYS on **TAPEIN** and **TAPEOUT** indicates that both the input and output should be handled by an ExHPDM server stream task. Note the use of WAIT(NO) on the TAPEIN and TAPEOUT DDs. This prevents other processing from being held up or a deadlock occurring when stream file resources for these connections cannot be obtained. The CLASS specification is being used on the TAPEOUT DD to bypass the need for a SELECT rule.

```
//COPYT      EXEC  PGM=FDRTCOPY,REGION=5M
//SYSPRINT   DD    SYSOUT=*
```

```
//TAPEIN   DD    SUBSYS=(SOV,'WAIT(NO)'),
//         DSN=OWPL.DUMP(0)
//TAPEOUT  DD    SUBSYS=(SOV,'CLASS(FDR) WAIT(NO)'),
//         DSN=OWPL.DUMP.COPY(+1)
//SYSIN DD *
  COPY MAXERR=1
```

Example 6

The following example shows the usage of FDRABR to perform a full volume dump of the volume TVOL01. The specification of DD SUBSYS on TAPE1 indicates that the FDRABR backup data set is made to ExHPDM. The DSN specification is not required as FDRABR overrides the data set name used on the DD to its own name format to identify generations and cycles of backups. TYPE=FDR indicates that a full volume backup is performed. This results in a new FDRABR generation being created. If TYPE=ABR was specified, then an incremental backup is performed. This would result in a new FDRABR cycle being created.

```
//DUMPABR  EXEC  PGM=FDRABR,REGION=0M
//SYSPRINT DD    SYSOUT=*
//TAPE1    DD    SUBSYS=SOV
//DISKVOL1 DD    VOL=SER=TVOL01,UNIT=SYSALLDA,DISP=OLD
//SYSPRIN1 DD    SYSOUT=*
//SYSIN    DD *
  DUMP TYPE=FDR,MAXERR=1
```

The following ExHPDM parameters may be specified in the startup parameter file for the above example.The selection rule (ABRRULE) can be used to relate the FDRABR dump data set to a particular ExHPDM stream class (FDR); this is, in turn, connected to an ExHPDM stream (FDR) through the class definition.

```
SELECT(ABRRULE PGM(FDRABR) CLASS(FDR))
CLASS(FDR STREAM(FDR)).
```

Example 7

The following example shows the usage of FDRCRYPT to perform a full volume dump of the volume VTPE01 and to encrypt the data onto tape. The specification of DD SUBSYS on TAPE1 indicates that the FDR backup dataset is made to ExHPDM.

```
//DUMPEXECPGM=FDR,REGION=5M,PARM='A'

//SYSPRINT DDSYSOUT=*

//SYSOUT  DDSYSOUT=*

//SYSTERM DD SYSOUT=*

//SYSUDUMP DDSYSOUT=*

//DISK1  DDUNIT=3380,DISP=OLD,VOL=SER=VTPE01

//TAPE1  DDSUBSYS=(SOV),

//DSN=BACKUP.BU1115A.FDR.VTPE01,

// DISP=(,KEEP),VOL=(,,,20)

//SYSPRIN1 DDSYSOUT=*

//SYSIN DD *

 DUMP TYPE=FDR,ENCRYPT=ALL,ENCRYPTTYPE=CIPHER

//FDRCRYPT DDDSN=FDRCRYPT.OPTIONS,DISP=SHR
```

Exploiting Parallelism with ExHPDM

ExHPDM works most efficiently with FDR when the FDR directives are processed in parallel ("*simultaneously*") instead of serially. By running FDR directives in parallel ExHPDM is able to interleave and stream the data efficiently to and from its stream files. FDR directives may be run in parallel in a number of ways depending on the type of FDR processing.

FDR and FDRDSF

Parallel processing for FDR programs FDR and FDRDSF (PGM=FDR or PGM=FDRDSF) can be performed in the following ways:

- With multiple jobs, where each job has a unique jobname. A single TAPE*x*/DISK*x* pair being allocated to each job, or
- For DUMP processing by specifying multiple TAPE*x*/DISK*x* DD pairs and the FDR ATTACH or MAXTASKS operands. When the ATTACH directive is used, this effectively sets the MAXTASKS value to the number of TAPE*x* specifications, up to the maximum value of 9. In general, 6 to 9 parallel dump tasks can be specified per

FDR dump job, depending on the amount of 'below the line' private region available. The storage requirements for each dump task are documented in the FDR user documentation. To obtain parallelism for dump processing when more than 9 TAPE*x* DDs are required then multiple concurrent jobs must be run, or

- For RESTORE TYPE=FDR processing by specifying multiple TAPE*x*/DISK*x* DD pairs and the FDR MAXTASKS operand, up to the maximum value of 9. To obtain parallelism for restore processing when more than 9 TAPE*x* DDs are required then multiple concurrent jobs must be run, or
- For RESTORE TYPE=DSF processing by using multiple jobs, where each job has a unique jobname. A single TAPE*x*/DISK*x* pair being allocated to each job. Neither the FDR ATTACH or MAXTASKS operands may be used when performing data sets restores.

Caution: When restoring data sets using FDR RESTORE TYPE=DSF, then parallel restores should not be used if any of the restoring data sets are multi-volume. A parallel restore can result in multi-volume data sets being incorrectly restored.

FDRTCOPY and FDRTSEL

Parallel processing for FDR programs FDRTCOPY and FDRTSEL (PGM=FDRTCOPY or PGM=FDRTSEL) can only be performed with multiple jobs, where each job has a unique jobname.

FDRABR

Parallel processing for FDR program FDRABR (PGM=FDRABR) can be performed in the following ways:

- With multiple jobs, where each job has a unique jobname. A single TAPE*x* DD, and optionally DISK*xxxx* DD, being allocated to each job. Parallel processing for FDRABR restores always requires multiple jobs,

Caution: When performing data set restores using FDRABR (TYPE=ABR or TYPE=DSF) then parallel restores should not be used if any of the restoring data sets are multi-volume. A parallel restore can result in multi-volume data sets being incorrectly restored.

or

- For DUMP processing by specifying up to 9 TAPE*x* DDs. Each TAPE*x* DD specifies to FDRABR to run another parallel dump task; there is no ATTACH or MAXTASKS operand for FDRABR. This allows up to 9 parallel dump tasks. FDRABR can select a large group of devices to be dumped depending on the volume or data set selection criteria. Where FDRABR selects to process more volumes than there are TAPE*x* DDs, then FDRABR will control the starting of new dumps as currently executing ones complete. The number of volumes selected for processing by FDRABR should be carefully considered to allow the maximum number of parallel processes to be performed. In general, 9 TAPE*x* DDs should be specified per FDRABR dump job depending on the amount of 'below the line' private region available. The storage requirements for each dump task are documented in the FDR

user documentation. Multiple FDRABR jobs might be required to allow ExHPDM to stream data efficiently.

Examples

The following selection rules apply to Examples 1, 2, 3 and 5 only. These are used to relate the **.RD3A.FULL.** data set (FDRRULE) or the usage of the FDRABR program (ABRRULE) in the FDR ExHPDM processing to use particular stream classes (FDRCLAS and ABRCLAS); these classes are in turn connected to ExHPDM streams (FDRSTRM and ABRSTRM). Examples 4 and 6 do not require any selection rules.

```
SELECT(FDRRULE DSN(**.RD3A.FULL.**) CLASS(FDRCLAS))
SELECT(ABRRULE PGM(FDRABR) CLASS(ABRCLAS)

CLASS(FDRCLAS STREAM(FDRSTRM))
CLASS(ABRCLAS STREAM(ABRSTRM))

STREAM(FDRSTRM DSN(STREAM.DS) CONCURRENT(WRITE(1)))
STREAM(ABRSTRM DSN(STREAM.DS.T&&LHHMMSS) CONNECTIONS(5))
```

Example 1

In the following example, three full volume dumps are processed in parallel by running three FDR jobs. Their output is directed to the same stream task in ExHPDM.

```
//DUMPJOB1  JOB jobcard info
//* Job backs up 1 pack in parallel
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP001  EXEC  PGM=FDR,REGION=4M
//SYSPRINT DD  SYSOUT=*
//DISK1    DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//TAPE1    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//        DISP=(,CATLG,DELETE)
//SYSPRIN1 DD  SYSOUT=*
//SYSIN    DD  *
 DUMP TYPE=FDR,MAXERR=1
/*
//DUMPJOB2  JOB jobcard info
//* Job backs up 1 pack in parallel
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP002  EXEC  PGM=FDR,REGION=4M
//SYSPRINT DD    SYSOUT=*
//DISK1    DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//TAPE1    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
//        DISP=(,CATLG,DELETE)
//SYSPRIN1 DD  SYSOUT=*
//SYSIN    DD  *
 DUMP TYPE=FDR,MAXERR=1
/*
//DUMPJOB3  JOB jobcard info
//* Job backs up 1 pack in parallel
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP003  EXEC  PGM=FDR,REGION=4M
//SYSPRINT DD  SYSOUT=*
```

```
//DISK1    DD  UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//TAPE1    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM03,
//         DISP=(,CATLG,DELETE)
//SYSPRIN1 DD  SYSOUT=*
//SYSIN    DD  *
 DUMP TYPE=FDR,MAXERR=1
/*
```

Example 2

In the following example, the same three full volume dumps from the previous example are processed in parallel by FDR using the ATTACH directive and one job. Their output is directed to the same stream task in ExHPDM.

```
//DUMPJOB  JOB jobcard info
//* Job backs up 3 packs (in parallel)
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP001  EXEC  PGM=FDR,REGION=4M
//SYSPRINT DD  SYSOUT=*
//DISK1DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DISK2DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DISK3DD  UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//TAPE1    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//         DISP=(,CATLG,DELETE)
//TAPE2    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
//         DISP=(,CATLG,DELETE)
//TAPE3    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM03,
//         DISP=(,CATLG,DELETE)
//SYSPRIN1 DD  SYSOUT=*
//SYSPRIN2 DD  SYSOUT=*
//SYSPRIN3 DD  SYSOUT=*
//SYSIN    DD  *
 DUMP TYPE=FDR,ATTACH,MAXERR=1
/*
```

Example 3

In the following example, the same three full volume dumps are processed as was done in example 2. Instead of using the ATTACH directive this example is using the MAXTASKS directive.

```
//DUMPJOB  JOB jobcard info
//* Job backs up 3 packs (in parallel)
//* backups will be to a single stream file due to
//*                  concurrent(write(1))
//STEP001   EXEC  PGM=FDR,REGION=4M
//SYSPRINT  DD  SYSOUT=*
//DISK1    DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DISK2    DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DISK3    DD  UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//TAPE1    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//DISP=(,CATLG,DELETE)
//TAPE2    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
//DISP=(,CATLG,DELETE)
//TAPE3    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM03,
//DISP=(,CATLG,DELETE)
```

```
//SYSPRIN1 DD  SYSOUT=*
//SYSPRIN2 DD  SYSOUT=*
//SYSPRIN3 DD  SYSOUT=*
//SYSIN    DD  *
 DUMP TYPE=FDR,MAXTASKS=3,MAXERR=1
/*
```

Example 4

In the following example, three full volume restores are processed in parallel by FDR using the MAXTASKS directive and one job.

```
//RESTJOB  JOB jobcard info
//* Job restores 3 packs (in parallel)
//STEP001  EXEC  PGM=FDR,REGION=4M
//SYSPRINT DD  SYSOUT=*
//DISK1    DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DISK2    DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DISK3    DD  UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//TAPE1    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01
//TAPE2    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02
//TAPE3    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM03
//SYSPRIN1 DD  SYSOUT=*
//SYSPRIN2 DD  SYSOUT=*
//SYSPRIN3 DD  SYSOUT=*
//SYSIN    DD  *
 RESTORE TYPE=FDR,MAXTASKS=3,MAXERR=1
/*
```

Example 5

The following example shows the usage of FDRABR to perform parallel full volume dumps of a group of volumes starting with the volser TVOL. The specification of DD SUBSYS on the TAPE*x* DDs indicates that all the FDRABR backup data sets will be made to ExHPDM. As 9 TAPE*x* DDs have been specified, FDRABR will process up to 9 dumps in parallel. If more than 9 TVOL volumes exist then new dumps will be started when existing ones complete. The JCL DD DSN= specification is not required as FDRABR overrides the data set name used on the DD to its own name format to identify generations and cycles of backups. A new data set will be created for each volume involved in the backup. TYPE=FDR indicates that full volume backups will be performed. This results in new FDRABR generations being created. If TYPE=ABR was specified then incremental backups of each volume will be performed. This would result in new FDRABR cycles being created for each volume.

```
//DUMPJOBJOB jobcard info
//* Job backs up 9 packs (in parallel)
//* backups will be to 2 stream files due to
//*                  connections(5)
//DUMPABR   EXEC     PGM=FDRABR,REGION=0M
//SYSPRINT  DD       SYSOUT=*
//TAPE1     DD       SUBSYS=SOV
//TAPE2     DD       SUBSYS=SOV
//TAPE3     DD       SUBSYS=SOV
//TAPE4     DD       SUBSYS=SOV
//TAPE5     DD       SUBSYS=SOV
//TAPE6     DD       SUBSYS=SOV
```

```
//TAPE7    DD       SUBSYS=SOV
//TAPE8    DD       SUBSYS=SOV
//TAPE9    DD       SUBSYS=SOV
//SYSPRIN1 DD         SYSOUT=*
//SYSPRIN2 DD         SYSOUT=*
//SYSPRIN3 DD         SYSOUT=*
//SYSPRIN4 DD         SYSOUT=*
//SYSPRIN5 DD         SYSOUT=*
//SYSPRIN6 DD         SYSOUT=*
//SYSPRIN7 DD         SYSOUT=*
//SYSPRIN8 DD         SYSOUT=*
//SYSPRIN9 DD         SYSOUT=*
//SYSIN    DD *
 DUMP TYPE=FDR,MAXERR=1
 MOUNT VOLG=TVOL
```

Example 6

The following example shows the usage of multiple FDRABR jobs to perform parallel full volume restores for four of the TVOL volumes (TVOL01-TVOL04) which had been dumped in the previous example. The FDRABR DYNTAPE operand is used to allow FDRABR to dynamically allocate the required TAPE# DD in each job to the ExHPDM subsystem (SOV is the default used by FDRABR). When performing restores of many volumes which require ExHPDM it is recommended to put as many restore jobs into the system as possible to allow ExHPDM to efficiently stream the data.

```
//RESTJOB1  JOB jobcard info
//* Job restores 1 pack in parallel with other
//* RESTJOBx jobs.
//RESTABR  EXEC    PGM=FDRABR,REGION=0M
//SYSPRINT DD      SYSOUT=*
//DISK1    DD      VOL=SER=TVOL01,UNIT=SYSALLDA,DISP=OLD
//SYSIN    DD      *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=TVOL01
/*
//RESTJOB2  JOB jobcard info
//* Job restores 1 pack in parallel with other
//* RESTJOBx jobs.
//RESTABREXECPGM=FDRABR,REGION=0M
//SYSPRINTDDSYSOUT=*
//DISK1DDVOL=SER=TVOL02,UNIT=SYSALLDA,DISP=OLD
//SYSINDD*
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=TVOL02
/*
//RESTJOB3  JOB jobcard info
//* Job restores 1 pack in parallel with other
//* RESTJOBx jobs.
//RESTABR  EXEC   PGM=FDRABR,REGION=0M
//SYSPRINT DD     SYSOUT=*
//DISK1    DD     VOL=SER=TVOL03,UNIT=SYSALLDA,DISP=OLD
//SYSIN    DD     *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=TVOL03
/*
//RESTJOB4  JOB jobcard info
```

```
//* Job restores 1 pack in parallel with other
//* RESTJOBx jobs.
//RESTABR   EXEC   PGM=FDRABR,REGION=0M
//SYSPRINT  DD    SYSOUT=*
//DISK1     DD    VOL=SER=TVOL04,UNIT=SYSALLDA,DISP=OLD
//SYSIN     DD    *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=TVOL04
/*
```

FDR InstantBackup, HSDM and ExHPDM

The High Speed Data Mover (HSDM) is a unique product that facilitates fast data movement into and out of a StorageTek's Iceberg, Shared Virtual Array (SVA), and IBM's 9393 RVA (RAMAC Virtual Array). HSDM is integrated seamlessly with Innovation's Data Processing's FDR InstantBackup solution. When FDR InstantBackup is coupled with ExHPDM to backup “SNAPPED” data to tape, the improvements in performance and elapsed times is quite considerable.

In the following example, the snapshot target volumes at address 0100 and 0101 are part of a pool of targets and are used to snap volumes DATA01 and DATA02. When the job is ready to be submitted, the volumes may need to be quiesced. The FDRSNAP step copies all the data from the original volumes to the targets, and when it completes, updates to these volumes can be initiated. FDRDSF then backups these “snapped” volumes using ExHPDM to manage and control the two backups, processing to tape in parallel. When complete the storage associated with the “snapped” volumes is released.

Example 7.

```
//SNAP     EXEC PGM=FDRSNAP
//SYSPRINT DD SYSOUT=*
//TAPE1    DD DUMMY
//SYSIN    DD *
SNAP TYPE=FDR
MOUNT VOL=DATA01,SNAPUNIT=0100
MOUNT VOL=DATA02,SNAPUNIT=0101
//BACKUP   EXEC PGM=FDRDSF,REGION=2M,COND=(0,NE,SNAP)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DISK1    DD UNIT=SYSALLDA,VOL=SER=DATA01,DISP=OLD
//TAPE1     DD DSN=BACKUP.VDATA01(+1),UNIT=TAPE,DISP=(,CATLG),SUBSYS=SOV
//SYSPRIN1 DD SYSOUT=*
//DISK2    DD UNIT=SYSALLDA,VOL=SER=DATA02,DISP=OLD
//TAPE2     DD DSN=BACKUP.VDATA02(+1),UNIT=TAPE,DISP=(,CATLG),SUBSYS=SOV
//SYSPRIN2 DD SYSOUT=*
//SYSIN    DD *
DUMP TYPE=DSF,ATTACH,SNAP=(USE,REL),DCT=YES
SELECT DSN=DATA**
```

ExHPDM Administration Utility (SOVADMN)

The ExHPDM Administration utility, SOVADMN, is used in batch or dynamically invoked from an application program to perform the following:

- Database Processing
- Expiration of client files
- Stream volume processing
- Recover specific client data sets for disaster recovery

The ExHPDM Administration utility is only available when the ExHPDM server is active in the current MVS. The server is specified as a subsystem name through the JCL parm SSNAME(*xxxx*) value or through the JCL SNAM*xxxx* DD DUMMY statement. Where *xxxx* is the ExHPDM subsystem name.

The ExHPDM actions are specified using utility directives that are specified through the SYSIN file.

Examples

Example 1

The following example shows SOVADMN being invoked by standard JCL to connect to the ExHPDM server whose MVS subsystem name is SOV:

```
//ADMNLST  EXEC PGM=SOVADMN,PARM='SSNAME=SOV'
//STEPLIB  DD DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*  used for diagnostic messages.
//SYSIN    DD *
 ADMIN DATABASE LIST CLIENT(DSN(OWWT.**))
 ADMIN DATABASE LIST CLIENT(OWNER(OWWT) DSN(**))
/*
```

Example 2

In the following example, the presence of the JCL DD statements SNAMSOV, SNAMSOV1, and SNAMSOV2 causes the utility to connect to the first ExHPDM server that is active whose MVS subsystem names are SOV, SOV1, or SOV2:

```
//ADMNLST    EXEC PRM=SOVADMN
//STEPLIB    DD DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT   DD SYSOUT=*
//SYSTERM    DD SYSOUT=*  used for diagnostic messages.
//SNAMSOV    DD DUMMY   directs requests to SOV server if active
//SNAMSOV1   DD DUMMY   otherwise directs requests to SOV1 server
//SNAMSOV2   DD DUMMY   otherwise directs requests to SOV2 server
//SYSIN      DD *
 ADMIN DATABASE LIST CLIENT(DSN(OWWT**))
 ADMIN DATABASE LIST CLIENT(OWNER(OWWT) DSN(**))
/*
```

Database Processing

ExHPDM database administration can be performed using the SOVADMN utility as follows:

- List database elements
- Delete database elements
- Client file expiration
- Backup current database and journal data set
- Restore a previous database and apply the journal data set(s)

Disaster Recovery of Client Data Sets

The SCANSTReamfile command allows a ExHPDM server started in disaster recovery mode (DR) to recover stream file information so that specific client data sets may be processed by ExHPDM client jobs. This command is used when the ExHPDM database is inaccessible or not available in a disaster recovery environment.

Directing Utilities through the SYSIN File

The batch utility actions are specified through the SYSIN file. The commands are categorized by:

- DATABASE for database processing
- SCANSTReamfile for disaster recovery processing
- SCANEXPIRE for deleting expired client files

Any of the above may be invoked once or more in the SOVADMN job.

ExHPDM Server

Each defined ExHPDM MVS subsystem on a single MVS system can support a single ExHPDM server. An ExHPDM server complex is defined by the sharable ExHPDM database that maintains all of the ExHPDM managed resources. Any number of ExHPDM servers can share this database in either a single or multi–image environment. If more than one server is on a single MVS image, then each of the servers requires a unique subsystem definition.

The ExHPDM server must be active for clients to write to or read from ExHPDM managed stream files. When ExHPDM is inactive, all connection requests are rejected. You should start ExHPDM at the MVS system IPL and leave it active for the life of the IPL. However, ExHPDM can be shut down and restarted as required.

At startup, the ExHPDM server reads a configuration parameter file (SOVPRM*xx*) to determine how it is intended to run. The parameter file determines:

- database resource definition

- Log file requirements
- Command and message prefixes
- System security (SAF) requirements
- Tape Management System requirements
- Streaming requirements.

As shown in Figure 2. on page 13, the ExHPDM server consists of a director, a number of managers, and a dynamic set of stream tasks.

- The director and the managers are active for the life of the ExHPDM server.
- The stream tasks are started as connections are made by client jobs and are terminated when all client connections using that stream task are closed.

For write requests clients are connected to the ExHPDM server through the DD SUBSYS directive and the selection rules contained with the startup parameter file. The parameter file options and the processing order for identifying and connecting a client are as follows:

1. Identify a client connection for ExHPDM processing
2. Identify the performance attributes for the client connection
3. Identify the management attributes for the client data set
4. Identify the stream task for the client connection
5. Identify an output device to service the stream task.

This release of ExHPDM contains support for SMC V6.2 to set dynamically via SMC policies, DD SUBSYS during allocation of client datasets. This provides the ability for customers to without having to alter existing backup/restore jobs the ability to direct particular batch workloads to ExHPDM. Two additional parameters to SMC for IDAX policies have been developed, these are IDAXSUBSYS and IDAXPROG. IDAXSUBSYS allows for the specification of an ExHPDM subsystem name to be set, which is then dynamically set into the allocation request when the dataset is allocated by MVS. Also, the IDAXPROG specification provides the ability for the JOB Step program name to be changed to SOVDSSU from ADRDSSU for DFDSS batch jobs to be directed to ExHPDM.

Once these SMC policy directives have been defined, and a job is submitted, then the processing order described above for identifying and connecting a client continues as normal. For additional information of these SMC policy parameters, refer to the *SMC 6.2 Configuration and Administration Guide*.

For read requests, clients are connected to the ExHPDM server using the stream name that was used during the write processing. The current settings for the stream are used as defined in the startup parameter file. The same determination of an output device for a read stream is performed as noted above in “Identify an output device to service the stream task.”

An ExHPDM server can be started in disaster recovery mode (DR) when the ExHPDM database is unavailable or not accessible. Disaster recovery mode allows specific client data sets to be restored for use by client restore jobs.

Identify a Client Connection for ExHPDM Processing

In the first instance, a client job identifies its intended connections to ExHPDM through the DD SUBSYS directive in its JCL, via SMC IDAX policies where the DD SUBSYS parameter is dynamically set during allocation of the client dataset, and/or ExHPDM connection rules. These are also known as selection rules.

Note: Stream connection rules are only used for client write requests. They are not required for client processing to read from a stream file because ExHPDM already knows how to process the data set as defined by the original client write request.

The SELECT keyword in the ExHPDM parameter file identifies, through a number of criteria, the client data sets that may be connected to the ExHPDM server for write requests. You can specify any number of these rules. These are known as SELECT definitions.

An example of selection rules or SELECT definitions is as follows:

```
SELECT(SYSPROG CLASS(MVS) DSN(SYSP.**))
SELECT(APPL1 CLASS(APPL) DSN(PAYROLL.GROUP*.**) MGMT(APPL))
SELECT(APPL2 CLASS(APPL) PGM(FDR))
SELECT(APPL3 CLASS(APPL2) JOB(APPL) MGMT(APPL))
SELECT(NOGO EXCLUDE USER(OWPL))
```

The rule matching process starts from the first SELECT definition entry and uses the first matching selection rule to determine what to do with the connection.

The outcome of the selection rule processing is either a ExHPDM CLASS or a condition where a connection should not be made. When a CLASS is specified it defines the performance attributes of the connection and the streams that it can use. A condition where a connection should not be made is indicated by the SELECT EXCLUDE keyword. If DD SUBSYS was not specified, the job continues as if ExHPDM was not involved in the connection. If DD SUBSYS was specified then the client connection is failed. If no matching rules can be located for a connection then this is treated as an EXCLUDE.

If the connection is accepted then the SELECT statement can also specify the desired ManaGeMenT for the client data set. ManaGeMenT is ignored when specified on SELECT EXCLUDE statements.

Client DD SUBSYS JCL parameters can bypass the requirement for the selection rule through the use of the CLASS keyword. For example:

```
//PLDDOUT  DD SUBSYS=(SOV,'CLASS(DB2)'),. . . . .
```

If the CLASS is obtained from the client DD SUBSYS JCL parameters, then this also means that ManaGeMenT will not be obtained from a selection rule. However, ManaGeMenT can also be specified on CLASS or on the DD SUBSYS JCL parameters.

The DD SUBSYS JCL parameters can bypass the use of the selection rule ManaGeMenT keyword. For example:

```
//PLDDOUT  DD SUBSYS=(SOV,'ManaGeMenT(KEEP10)'),. . . . .
```

The DD SUBSYS JCL parameters are described in "DD SUBSYS JCL Parameter Examples" on page 280.

The SELECT keyword syntax is described in further detail in "SELECT Keyword" on page 134.

Identify the Performance Attributes for the Client Connection

The performance attributes for a write client connection are known as stream classes and are specified on the CLASS keyword. The CLASS is determined by the SELECT CLASS keyword or DD SUBSYS parameters discussed in the previous section "Identify a Client Connection for ExHPDM Processing" on page 36. The CLASS indicates to ExHPDM the type of data defined by the connection and the streams that the connection can use.

Note: Stream classes are only used for client write requests. They are not required for client processing to read from a stream file because ExHPDM already knows how to process the data set because of the original client write request.

CLASS is used in two contexts. The first context is in the parameter file, where each of the available set of stream class parameters is given a name and a definition. The second context is within the confines of individual SELECT or DD SUBSYS parameters. In the second case only the class name is specified, in which case the name refers to a set of class parameters defined in the parameter file.

Use of the CLASS keyword in the first context is known as 'definition'. In the second context it is known as 'reference', since the class name refers to a definition in the parameter file.

An example of a CLASS definition is as follows. The name of the CLASS is DB2:

```
CLASS(DB2 EFI(3.5M) MGMT(KEEP10) STREAM(DBB21 DB22))
```

An example of a CLASS reference in the DD SUBSYS JCL parameters is as follows:

```
//TAPE1  DD SUBSYS=(SOV,'CLASS(DB2)'),. . . . .
```

An example of a CLASS reference in a SELECT rule is as follows:

```
SELECT(DB2FILES CLASS(DB2) DSN(**))
```

Since a class name reference may be contained in any or all of a SELECT definition in the parameter file, or in the client DD SUBSYS parameters, there must be a rule used to determine which of these two definitions is actually used. This rule is based on specificity; the most specific reference overrides any less specific reference.

The order from most specific to least specific is as follows:

- DD SUBSYS

- SELECT.

The first CLASS reference encountered in this hierarchy is used to select the CLASS definition so named. If no class definition of this name is found, this is reported as an error. Otherwise, the selected CLASS definition is used for the client write request.

The expected performance level of a connection using the CLASS keyword is defined by the CLASS ExpectedFlowIncrease keyword. This parameter defines the expected output rate increase that the connection could generate. ExpectedFlowIncrease may be used to allow load balancing to be performed at the stream task level. This is explained in " Identify the Stream and Stream Task for the Client Connection" on page 41.

The management of the connecting data set using the CLASS keyword is defined by the CLASS ManaGeMenT keyword. This parameter defines the way in which the data set is to be managed by ExHPDM. If the ManaGeMenT parameter is also specified on the SELECT ManaGeMenT keyword or the DD SUBSYS JCL parameters, then this overrides the value specified in CLASS.

The stream class indicates a list of possible streams that the connection can use as defined through CLASS STREAM keyword. ExHPDM uses each stream in the specified list to locate an applicable stream task that will process the output for the client connection. The first specified stream that can successfully process the connection is the one that is used.

Identify the Management Attributes for the Client Data Set

The management attributes for a write client data set are known as the stream management which is specified on the ManaGeMenT keyword. The stream management name is determined by the SELECT ManaGeMenT keyword or DD SUBSYS parameters discussed in the section "Identify a Client Connection for ExHPDM Processing" on page 36 or the CLASS ManaGeMenT keyword discussed in the previous section "Identify the Performance Attributes for the Client Connection" on page 37.

Note: The ManaGeMenT for a client is only determined for client write requests. The ManaGeMenT is not required to perform a client read request.

The ManaGeMenT indicates to ExHPDM the way that the client data should be managed by ExHPDM. Essentially it determines the conditions under which client data sets should be expired.

ManaGeMenT is used in two contexts. The first context is in the parameter file, where each of the available set of management parameters is given a name and a definition. The second context is within the confines of individual SELECT, CLASS or DD SUBSYS parameters. In the second case only the management name is specified, in which case the name refers to a set of management parameters defined in the parameter file.

Use of the ManaGeMenT keyword in the first context is known as 'definition'. In the second context it is known as 'reference', since the management name refers to a definition in the parameter file.

An example of a ManaGeMenT definition is as follows. The name of the management definition is KEEP10:

```
MGMT(KEEP10 EXPIRE(RETAIN(10)))
```

An example of a ManaGeMenT reference in the DD SUBSYS JCL parameters is as follows:

```
//TAPE2  DD SUBSYS=(SOV,'MGMT(KEEP10)'),. . . . . .
```

An example of a ManaGeMenT reference in a SELECT rule is as follows:

```
SELECT(DB2FILES CLASS(DB2) MGMT(KEEP10) DSN(**))
```

An example of a ManaGeMenT reference in a CLASS is as follows:

```
CLASS(DB2 STREAM(DB2) MGMT(KEEP10))
```

Since a management name reference may be contained in any or all of a SELECT or CLASS definition in the parameter file, or in the client DD SUBSYS parameters, there must be a rule used to determine which of these three definitions is actually used. This rule is based on specificity; the most specific reference overrides any less specific reference.

The order from most specific to least specific is as follows:

- DD SUBSYS
- SELECT
- CLASS.

The first management reference encountered in this hierarchy is used to select the management definition so named. If no management definition of this name is found, this is reported as an error (message SOV06080E). Otherwise, the selected management definition is used for the client write request.

If a management reference is provided in the DD SUBSYS parameters, it is verified as a valid management reference for both read and write requests. Apart from this case, management references are ignored for client read requests.

When a management definition is determined, it is used as a source of expiration information for the client file. Expiration information can also be specified in the client DD statement, via the JCL parameters EXPDT or RETPD, in which case the JCL parameters override the management definition, except when the management class specifies WHENUNCAT with EQ99000 and EXPDT=99000 is coded in the JCL.

Client file expiration parameters are not passed on to MVS allocation. They are retained in the ExHPDM database record for the client file, and used exclusively by ExHPDM for automatic management of the client file.

The expiration parameters are interpreted as follows:

- If the parameters originate from the client JCL as EXPDT=1999/365 or EXPDT=1999/366, the client is not expired. This is for compatibility with existing JCL semantics. This rule is not applied if expiration is obtained from management definition parameter EXPiryDaTe. In this case the date is taken literally.

- If JCL parameter RETPD, or management definition parameter RETainPerioD, is specified, then the retain period is converted to a target date for expiration by adding the specified amount (in days) to the current time, and rounding up to the next whole number of days.

The following processing only applies if expiration parameters are obtained from a management definition:

- If expiration is obtained from management definition parameter EXPiryDaTe, the specified value is converted to a valid date e.g., 2000/900 is converted to 2002/169.
- If both RETainPerioD and EXPiryDaTe are specified, the later date is stored in the ExHPDM database record for the client.
- The SAVEGens and WhenUNCATaloged information is also stored in the database record.

Note: The name of the management definition (the management reference) is not stored in the client record in the ExHPDM database; only the information in the definition is stored. This means that changing a management definition will not change the expiration parameters of any client files which made use of the definition before its change.

After the expiry information is stored in the database, it is not accessed by ExHPDM until an ADMIN SCANEXPIRE job is run. At that time, if any, client files will be examined to see whether they have expired according to the original criteria. Since there are several criteria that may be used simultaneously, the following rules are used to decide whether the client file is expired:

- If the original management definition specified WhenUNCATaloged and the client file is not currently cataloged, then the file is deemed to be expired, regardless of any of the following.
- If the original management definition specified SAVEGens(number of gens), and the client file has 'number of gens' (or more) more recent generations, then the file is deemed to be expired, regardless of any of the following.
- If there is a target expiration date for this client file, and the current date is greater than that date, then the client is deemed to be expired.
- Otherwise, the client has not expired.

In the case that a client file has no explicit expiry information obtained from a management reference, the client file may still be expired if its containing stream has a RETainPerioD definition.

A stream file with a RETainPerioD provides a fallback expiry for all clients in that stream file which did not have any management reference when they were initially written. This fallback expiry can be a useful facility, however any expiry specification in stream definitions must be carefully planned to avoid unexpected expiration of client files.

A stream file with a RETainPerioD does not mean that the stream file itself can expire. Stream files are not managed this way; rather, stream files are managed based on the client files they contain. A stream file either has at least one client (in which case it cannot be

deleted) or all its clients have been deleted (in which case it is eligible to be deleted). A client file which is expired is not deleted until an ADMIN SCANEXPIRE or ADMIN DB DELETE command is run.

If no expiry information for a client can be obtained from the client JCL, the various ManaGeMenT keywords or the STREAM RETainPerioD then it is not eligible for expiration. That is, the default is to never expire client data sets.

The ManaGeMenT keyword syntax is described in further detail in "MANAGEMENT Keyword" on page 121.

Identify the Stream and Stream Task for the Client Connection

When a client connection is assigned a stream class, the stream that will process the write request is selected. The stream class is selected as described in the section "Identify the Performance Attributes for the Client Connection" on page 37.

Note: The stream is selected using the selection rules and stream classes for client write requests only. The stream name selected by this processing is stored in the ExHPDM database and is later used when performing the client read processing.

A stream definition is described by the STREAM keyword as contained in the ExHPDM startup parameter file. This keyword defines the attributes of the stream and its associated stream tasks. The stream is referenced by one or more stream classes that may use the facilities of the stream.

An example of a STREAM definition is as follows. The name of the STREAM is DB22:

```
STREAM(DB22 DSN(DB2.DUMP) DEVICE(T10000) CONNECTIONS(10)
CONCURRENT(WRITE(10)))
```

An example of a STREAM reference in the stream CLASS is as follows:

```
CLASS(DB22 EFI(3.5M) MGMT(KEEP10) STREAM(DB22))
```

The STREAM defines the attributes of a stream task. The stream task is the actual unit of work that processes connections. An occurrence of a stream task uses one or more input or output devices and reads or generates a single file called a stream file. A stream file contains data from all of the stream task client connections.

The devices used by the stream task are obtained from the list of supplied DEVICE definitions. The first device definition in the list that has not exceeded its limits will be used to satisfy the allocation request for the stream file. If there are no available devices in the device list then any new connections requesting the stream definition will wait unless the WAIT(NO) keyword is specified on the DD SUBSYS parameter or on the STREAM definition.

The STREAM keyword contains limits that indicate:

- the maximum number of parallel connections per stream task. This is indicated by the STREAM **CONNECTIONS** keyword.

- the maximum number of concurrent (parallel) stream tasks. This is indicated by the STREAM **CONCURRENT** keyword.
- an implied flow limit for the device being used by a stream task.
- optional selection of stream tasks by jobname, user name, or dsn level.

A new stream task is started in the following circumstances:

- no streams are active with the specified name or selection criteria.
- For read streams there is no stream task active for the required stream file name.
- For write streams:
  - the maximum number of parallel connections was reached in all currently active stream tasks with the same name.
  - the flow limit of the device being used by the stream task was reached or exceeded.

When the number of stream tasks has exceeded the limit specified on the STREAM **CONCURRENT** keyword, new connections requesting the stream definition will wait unless the **WAIT(NO)** keyword is specified on the DD SUBSYS parameter or on the STREAM definition.

The stream definition can indicate a list of the stream devices that can be used by the stream tasks. A list is allowed as stream devices may have limits associated with them. The first device definition that can be used by a stream task is the one that is chosen. If no device is specified, the device definition specified by the **DEFAULT** parameter will be used. If no **DEFAULT** device definition was specified then an implied device of **CART** is used. See “DEVICE Keyword” on page 107.

The stream is already known for a request to read from a tape volume because it was derived by the original write request processing and stored in the ExHPDM database.

The STREAM keyword syntax and stream tasks are described in further details in “STREAM Keyword” on page 142.

Identify an Input or Output Device to Service the Stream Task

The device definition describes the read or write devices available to a stream task. Streams and stream tasks were described in the previous section “Identify the Stream and Stream Task for the Client Connection” on page 41.

For read stream tasks, if no input device can be allocated by going through the DEVICE statements in the parameter file, then ExHPDM will attempt to allocate a device using the original generic unit name the stream file was created, such as 3490.

A device definition is described by the DEVICE keyword as contained in the ExHPDM startup parameter file. The device definition specifies as a list of generic device types or esoteric names for a device group.

An example of a device definition is as follows:

```
DEVICE(REDWOOD UNIT(REDDD31) DEVL(10))
```

An example of a device reference in a STREAM is as follows:

```
STREAM(FAST DSN(FAST.DUMP) DEVICE(T10000))
```

The DEVICE DEViceLimit keyword specifies the maximum number of devices that can be used by ExHPDM. When the DEViceLimit is reached, then no further devices can be used in that device definition. If there are any other devices listed in the stream definition then it attempts to use one of these.

The maximum performance that can be achieved for the device can be specified by the DEVICE **FLOWLIMIT** keyword. For example, the following device definition indicates that the devices contained in the esoteric 9840 have a maximum performance of 10 MB per second.

```
DEVICE(FASTDEV UNIT(9840) FLOWLIMIT (10M))
```

When all connections to a device are closed for a stream task, the device is freed unless the DEVICE **RETAIN** keyword is specified. The **RETAIN** parameter is supplied at the device level because some devices are more valuable than others and must be released quicker to allow other units of work to proceed. A different RETAIN period may be specified for READ or WRITE requests. In addition, the RETAIN period for the device may be modified, and is overridden, by the RETAIN specification on the DD SUBSYS parameters.

A default device entry can be supplied to direct stream tasks that do not have a device list to a default device through the use of the DEVICE **DEFAULT** keyword. If there is no DEFAULT device specified then an internal default is generated as follows:

```
DEVICE(DEFAULT UNIT(CART) DEVL(16) RETAIN(WRITE(0) READ(0)) UNITCNT(1))
```

The DEVICE keyword syntax is described in further detail in “DEVICE Keyword” on page 107.

The Relationship of Stream Parameter Definitions

The method used to group clients to particular streams is performed through selection rules, classes, and streams.

As shown in Figure 2. on page 13, the ExHPDM server consists of a number of stream tasks. An individual occurrence of a stream task has one stream. The ExHPDM server can have any number of stream tasks active concurrently with different stream names or the same name up to a limiting value specified by the **CONCURRENT** parameter of the STREAM keyword. Some of these stream tasks may be reading tasks and some may be writing tasks. However, a single stream task cannot read and write.

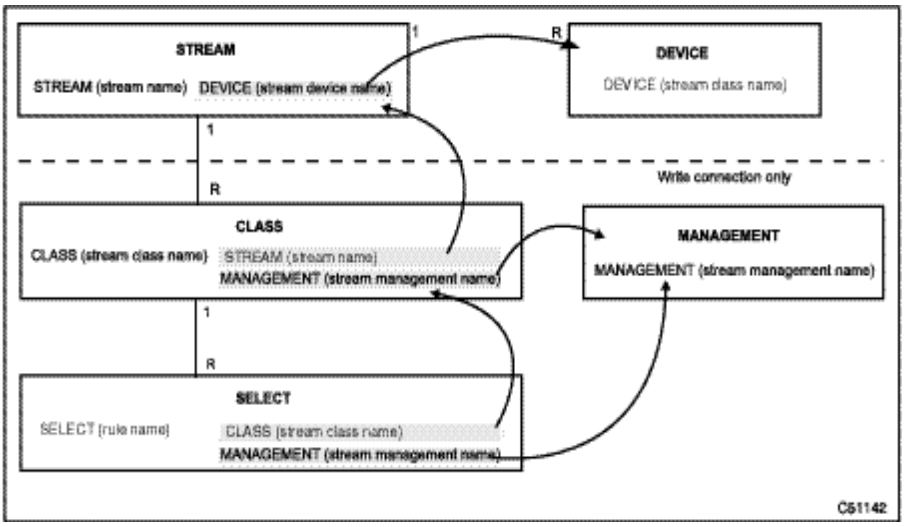


Figure 3. Stream Parameter Relationships

Figure 3. shows the relationship between the various stream parameter definitions. This figure is from the perspective of a single stream definition.

The grouping of client data to a stream definition is performed by:

- the CLASS keyword for connections that are writing, or
- the volumes on which the data resides for read operations.

The relationship of each of the levels is shown by the shaded parameters in Figure 3. along with the linking arrows in each of the boxes. There are other keywords that are relevant to each of these, which are described in more detail by the syntax that can influence this relationship. Some of these parameters may also be specified on the clients DD SUBSYS. The DD SUBSYS values override those in the parameter file.

A stream definition can only use a single device definition as specified by the DEVICE keyword. However, the determination of the stream device definition may be specified in a list. This list is created so you can specify the maximum number of allocations per device unit as specified by the DEVICE keyword.

The specifications of the CLASS and SELECT keywords are only used during write processing.

During read processing, the same STREAM parameter of the STREAM keyword of the write processing is used. The STREAM(*stream name*) parameter is stored in the database for the connection object (the DSN(*stream file name*) parameter of the STREAM keyword). Values specified on the most recently generated parameters for the STREAM keyword are used. However, the stream device definition is selected based on what is specified in the current DEViceLimit parameter of the DEVICE keyword.

The stream file contains additional JCL style parameters that specify the data set name of the stream file, its disposition, expiration date, or retention period.

The stream file can be optionally cataloged in the MVS catalog.

The stream file can be optionally expired and automatically deleted after a period of time as specified by the **RETainPerioD** parameter of the STREAM keyword. A stream file will not be deleted when it contains clients that have not yet reached their expiration date. Stream files that have reached or exceeded their retain period which do not contain non expired client data sets will be deleted when the SOVADMN directive ADMIN SCANEXPIRE is used. See "SCANEXPIRE" on page 265.

Warning: The STREAM EXPDT parameter should not be used to expire a stream file. EXPDT should only be used to allow TMS triggers to occur when a stream file is deleted by ExHPDM. See "Using Expiration Date Processing triggers" on page 52.

Abnormal Termination of Client Connections

Any processing errors detected by the ExHPDM server while processing a client request will yield a logical I/O error and/or an error message for these connections. Additional ExHPDM messages are directed to the client's JESYSMSG or job journal data set and also to the ExHPDM server log.

When client write processing is terminated before its processing is completed, the ExHPDM database only records a partially complete record of the client data set. The client data set is invalid and cannot be accessed for read processing. However, successfully completing client connections are unaffected by clients which have abnormally terminated. The output of the client jobs should be examined to determine which connections have successfully completed.

In all cases of abnormal termination of a connection, these should be rerun from the beginning. ExHPDM does not support any restart or checkpoint processing capability.

DFSMSdss

Any DFSMSdss client connection that is executing at the time that either the ExHPDM server address space abnormally or prematurely terminates, or the connection is refused or broken, receives a logical I/O error and an error message for its current connection(s). This will cause those DFSMSdss functions to terminate.

Depending on the type of DFSMSdss processing and where the termination occurs one of the following messages could be output by DFSMSdss:

ADR356E (*xxx*)-*mmmmm*(*yy*), task terminated by UIM exit (*aa*)

Where *aa* is the UIM exit point at which the DFSMSdss function was terminated.

ADR374E (*xxx*)-*mmmmm*(*yy*), UNABLE TO OPEN DDNAME *ddname*, 10

These messages are also accompanied by ExHPDM messages in the client's JESYSMSG data set and to the ExHPDM log file to indicate what caused the failure. Messages are also

issued by ADRDSSU to its SYSPRINT data set to record the status of the DFSMSdss transaction.

FDR

Any FDR client connection that is executing at the time that either the ExHPDM server address space abnormally or prematurely terminates, or the connection is refused or broken, receives an open, close, or logical I/O error and an error message for its current connection(s). This causes those FDR functions to terminate with an FDR U0888 and possibly a subsequent FDR U0200 ABEND. The ABEND U0200 is normally generated when the termination occurs during I/O processing. The normal message that FDR outputs for the termination is:

FDR319 FDR OPERATION ABNORMALLY TERMINATED VOL=*vvvvvv* COMP CODE=S*ssss* U*uuuu*

Depending on the stage that the connection is broken, the following errors will be observed:

- Termination during open processing will result in an ABEND S013 with a reason code of C0
- Termination during close processing will result in an ABEND S614 with reason code of 10
- Termination during read(restore)/write(dump) processing will result in a logical I/O error

Note:

1. Termination during open processing generally occurs due to the connection being refused to the ExHPDM server.
2. When termination occurs during read (restore) / write (dump) processing, then in addition to a logical I/O error, the FDR job also gets an ABEND S614 with reason code of 10. This occurs during close processing. These type of errors normally occur if the client connection is cancelled or the server terminates.
3. FDR normally terminates with an ABEND U0200 after detecting 20 errors from its input or output. This occurs when ExHPDM attempts to terminate a connection during I/O processing. The FDR option MAXERR=1 may be specified on the FDR processing statements to terminate after receiving 1 error.

These errors are also accompanied by a message in the client's JESYSMSG data set and to the ExHPDM log file to indicate what caused the failure. Messages are also issued by FDR to its SYSPRINT and SYSPRIN*x* data sets to record the status of the FDR transaction.

System Security, Authorization, and Verification

This section describes the system associated with the software security aspects of ExHPDM.

All authorization and verification are performed by ExHPDM using services available through the MVS System Authorization Facility (SAF). Information available at the source of the transaction instance is used by ExHPDM to perform authorization and verification. All authorization and verification are performed using security information maintained and accessed by your MVS enterprise system.

Note: Access to the actual data sets that are being dumped or restored is still required through the use of the standard SAF mechanism. Standard DFSMSdss and FDR SAF issues apply.

ExHPDM performs resource verification on the following:

- DATASET profiles for each data set opened by the client connections
- FACILITY class or user chosen class profiles for verification of ExHPDM streaming resources.

Additionally, for the ExHPDM Administration Utility SCANSTReamfile command processing, the server is required to access stream files in BLP (Bypass Label Processing) mode. BLP can constitute special security authority.

If no security product is installed on your system, no verification is performed.

DATASET Profile Verification

DATASET profiles are validated by the ExHPDM server at the time of the client connection, generally whenever an open is performed. The profile used for the validation is the data set name specified on the allocation.

Note that this verification is required as SAF, and therefore RACF, does not get involved in subsystem allocation (DD SUBSYS) requests. However, some security products that implant hooks into the MVS operating system might duplicate the data set verification with ExHPDM.

ExHPDM Streaming Verification

ExHPDM streaming verification is performed at the STREAM and CLASS keyword level to ensure that a client connection is authorized to particular ExHPDM resources.

By default, the FACILITY class is used to contain all the ExHPDM profiles. However, another user-defined class can be used. This is indicated by the SAF keyword in the ExHPDM startup parameter file. You need to define this class to your security product. See “SAF Keyword” on page 133 for additional information.

The standard format of ExHPDM profile names is shown in the following table.

Table 1. ExHPDM Profile Names

Profile Name	Description
SOV.STREAM.*stream name*	Validates access to the specified stream. For details about streams, see “STREAM Keyword” on page 142.
SOV.CLASS.*stream class name*	Validates access to the specified stream class. For details about stream classes, see “CLASS Keyword” on page 99.

The following example of output displays the failed verification of a connection that specified a stream class that it was not authorized to use:

```
ICH408I  USER (OWWT   ) GROUP(IXFPAUS) NAME(WART)
SOV.CLASS.TIMCLASS CL(FACILITY)
INSUFFICIENT ACCESS AUTHORITY
ACCESS INTENT(READ   ) ACCESS ALLOWED(NONE   )
```

Using ExHPDM with Your Tape Management System

ExHPDM manages its own tape inventory by recording tape volume details in its own database. When ExHPDM requires a new volume for write stream processing, it requests this as a non specific, scratch, mount request. The volume information used to satisfy this request is located in the ExHPDM database. Consequently, any volumes used by ExHPDM should not be scratched without deleting the volume details from the ExHPDM database.

Once a volume is used to satisfy a non specific mount request for ExHPDM then that volume cannot be used to satisfy any further non specific mount requests without that volume first being removed from the ExHPDM database. ExHPDM will reject the mount and prompt for another volume to be mounted. ExHPDM does this to ensure that a previously recorded volume is not overwritten accidentally. The normal message that ExHPDM outputs is:

SOV06213W VOLSER *volume_serial_number* **already used by** *stream_file_name***. Mount another.**

ExHPDM volumes are made available for reuse through the SOVADMN utility.

Volumes that ExHPDM is no longer using can be returned to the TMS in two separate ways:

1. Interfacing directly to the TMS
2. Using expiration date processing triggers

Interfacing Directly to the TMS

ExHPDM can interface directly to any TMS to notify when a volume has become available. When interfacing with the TMS, ExHPDM is known as the External Data Manager (EDM) to the TMS. Currently ExHPDM has provided a direct interface as an EDM for the following TMSs:

- CA–1

- TLMS
- ASG-Zara (formerly AutoMedia)
- CONTROL–T
- DFSMSrmm
- TAPE2000.

Specifying ExHPDM as an EDM is tape management system dependent. If ExHPDM is identified as an EDM then review the usage of the STREAM **CATALOG** keyword. **CATALOG(NO)** should be specified for some TMSs to prevent warning messages from being displayed. Refer to "STREAM Keyword" on page 142 for more information.

Being identified as an EDM allows ExHPDM to advise the tape management system when it has finished with a tape volume. Initially, the tape management system passes control or management of the tape volume to ExHPDM when a stream file requests the mount of a non specific volume. ExHPDM owns the volume until a SOVADMN utility is issued to release the usage of the volume.

If ExHPDM is used as an EDM, it should be identified to the TMS by the program name of SOVMAIN. Additional EDM identification can be specified to the TMS, for example, stream file name (data set name masking) or address space name. You must consult the appropriate installation or customization manual for your tape management system for further details about identifying ExHPDM as an External Data Manager.

Note: ExHPDM can interface with other TMSs not listed by programming a scratch volume routine. Refer to "TMS Keyword" on page 155 for more information.

Specifying the TMS Keyword

The startup parameter file TMS keyword permits the identification of a TMS and a loadable routine that is invoked by the ExHPDM server for each volume that is scratched from a deleted stream file. See "TMS Keyword" on page 155" for more information.

Specifying a Loadable Routine

The **SCRatchVOLumeRouTiNe** parameter of the TMS keyword specifies the entry point of a serially reusable or reentrant routine that performs specialized tape volume scratch processing. The routine is loaded and called and deleted each time ExHPDM encounters a stream file tape volume that is to be scratched. This permits the load module to be refreshed at any time.

When a program error is encountered during processing in the routine called by the **SCRatchVOLumeRouTiNe** parameter, ExHPDM issues an error message to the MCS console indicating the routine has failed and disables the routine. No further calls are made to this routine until ExHPDM is restarted or a SET **PRM** operator command issued.

If a TMS is being used that is not one of those supported by ExHPDM then a **SCRatchVOLumeRouTiNe** may be coded. The TMS(USER SCRVOLRTN(*routine name*)) specification may be used to invoke this routine. The parameters passed to the **SCRatchVOLumeRouTiNe** are described by "TMS Keyword" on page 155.

Using CA–1 with ExHPDM

The CA–1 tape management system can be identified to ExHPDM by specifying the TMS keyword in the ExHPDM startup parameter file. ExHPDM will use the TMSTMSTV scratch routine as supplied by CA–1:

```
TMS(CA1)
```

To identify ExHPDM to CA–1 as an EDM the CA–1 parameter file member TMOEDM*xx* needs to be created/updated. For example, the following ExHPDM EDM is know as SOV although any name acceptable to CA–1 may be used:

```
EDM=SOV,PGM=SOVMAIN
```

Refer to the CA-1 user documentation for further details.

Using CONTROL–T with ExHPDM

The Control–T tape management system can be identified to ExHPDM by specifying the TMS keyword in the ExHPDM startup parameter file. ExHPDM will use the SOVCTTEX scratch routine as supplied by CONTROL–T specifically for use by ExHPDM:

```
TMS(CONTROLT)
```

The Control–T PTF FI05302 for levels 5.0.0, 5.0.4 and 5.1.4 are required to support ExHPDM.

The Control–T specification rules DO RETENTION=EDM is required for data sets created by ExHPDM. This statement indicates that the data sets indicated in the rules selection criteria are EDM controlled. It is recommended that ON PGM=SOVMAIN is used as the selection criteria in the rules.

The Control–T user documentation contains specific instructions on setting up and using ExHPDM as an EDM.

Using DFSMSrmm with ExHPDM

The DFSMSrmm tape management system can be identified to ExHPDM by specifying the TMS keyword in the ExHPDM startup parameter file. ExHPDM will use the EDGDFHSM scratch routine as supplied by DFSMSrmm:

```
TMS(RMM)
```

Define ExHPDM to the DFSMSrmm SAF resources. Without this the EDGDFHSM release will not work:

- If STGADMIN.EDG.RELEASE is defined, then give ExHPDM read access to STGADMIN.EDG.RELEASE.
- If STGADMIN.EDG.RELEASE is not defined, then give ExHPDM read access to STGADMIN.EDG.MASTER.
- Define STGADMIN.EDG.OWNER.*exhpdmid* for each ExHPDM server USER id and give the other ExHPDM server USER ids UPDATE access to it. This allows one

ExHPDM server to release the tapes initially obtained from scratch by another ExHPDM server.

Update the DFSMSrmm parameter member EDGRMM*xx* (or via RMM TSO/ISPF commands) as follows:

- Include a Vital Record Specification (VRS) for ExHPDM stream files. This must be set so that the VRS delete date is 'never delete' (i.e 1999/365) and the COUNT set to never expire the stream file. For example:

  RMM ADDVRS DSNAME('') JOBNAME(*ExHPDM proc*) COUNT(1) CYCLES DELETEDATE(1999/365)**

  Note that COUNT(1) may be specified since each ExHPDM stream file is unique. This will ensure that the stream files are forever retained until released by ExHPDM.

  This VRS is done by jobname but could just as well be done for DSNAME so that different LOCATION information may be specified for differing streams. Additional VRS may be set to define LOCATION for volume rotation.

- Add abnormal termination VRS specification to retain stream files in the extremely unlikely circumstance of an ExHPDM server abend. This must be done as ExHPDM still has a stream file recorded in its database along with the volumes that it has used up to the time of the abend. For example:

  RMM ADDVRS DSNAME('ABEND') JOBNAME(*ExHPDM proc*)
  COUNT(1) CYCLES DELETEDATE(1999/365)
  RMM ADDVRS DSNAME('OPEN') JOBNAME(*ExHPDM proc*)
  COUNT(1) CYCLES DELETEDATE(1999/365)

Refer to the *DFSMS/MVS DFSMSrmm Implementation and Customization Guide* for further details.

Using TAPE2000 with ExHPDM

The TAPE2000 tape management system can be identified to ExHPDM by specifying the TMS keyword in the ExHPDM startup parameter file. ExHPDM will use the SOVT2TEX scratch routine as supplied by TAPE2000 specifically for use by ExHPDM:

TMS(TAPE2000)

The default TMS subsystem for this specification is TLFS. If a different subsystem is being used then this must be specified in the TMS SSNAME keyword.

The default TMS EDM name as defined to TAPE2000 is SOV. If a different EDM name is being used then this must be specified in the TMS DataManagerNAME keyword.

ExHPDM must be identified to TAPE2000 as an EDM. Refer to the TAPE2000 user documentation for details.

Note: Contact StorageTek technical support for the availability of the ExHPDM support in TAPE2000.

Using TLMS with ExHPDM

The TLMS tape management system can be identified to ExHPDM by specifying the TMS keyword in the ExHPDM startup parameter file. ExHPDM will use the TLMSSEDM scratch routine as supplied by TLMS:

```
TMS(TLMS)
```

To identify ExHPDM to TLMS as an EDM the TLMS parameter file member CTOEDM*xx* needs to be created/updated. For example, the following ExHPDM EDM is know as SOV although any name acceptable to TLMS may be used:

```
EDM=SOV,PGM=SOVMAIN
```

Refer to the TLMS user documentation for further details.

Using ASG-Zara (formerly AutoMedia) with ExHPDM

The ASG-Zara tape management system can be identified to ExHPDM by specifying the TMS keyword in the ExHPDM startup parameter file. ExHPDM will use the SOVAMTEX scratch routine as supplied by ASG-Zara specifically for use by ExHPDM:

```
TMS(ZARA)
```

The default TMS subsystem for this specification is ASG-Zara. If a different subsystem is being used then this must be specified in the TMS SSNAME keyword.

ASG-Zara (formerly AutoMedia) needs to be at tape level T130A036, or higher, to support ExHPDM.

ASG-Zara (formerly AutoMedia) must be set up to assign ExHPDM created tapes an expiration of US002.

The ExHPDM started task name can be used in the jobname field of an ASG-Zara Expiration Candidate Table entry to give ExHPDM created volumes the correct expiration value. The value of the ASG-Zara generation option EXPDT TABLE (Default/Override) will not matter so long as ExHPDM is not told to use any particular expiration date on its stream files as specified of STREAM EXPDT. If an expiry date is specified through ExHPDM, the ASG-Zara EXPDT TABLE option must be set to Override to have a US002 expiration assigned.

The ASG-Zara user documentation contains specific instructions on setting up and using ExHPDM as an EDM.

Using Expiration Date Processing triggers

An alternative to interfacing directly with a TMS is to use expiration date triggers. An expiration date (EXPiryDaTe parameter) may be specified on the STREAM keyword to cause TMS specific processing to be performed at stream allocation time and/or stream deletion time. The use of an expiry date to cause these types of TMS triggers is TMS specific. For details on what triggers the TMS allows refer to the TMS documentation.

In most TMSs STREAM(EXPDT(1999/000)) can be used to trigger catalog control. This would enable volumes associated with the stream file to be returned as available to the TMS when the stream file is deleted and the ICF catalog entry is removed. To allow for this STREAM(CATALOG(YES)) should be specified or defaulted in this instance.

To prevent stream files from being expired, STREAM(EXPDT(1999/365)) or STREAM(EXPDT(1999/366)) may be used.

Warning: STREAM EXPDT should only be used for TMS triggers and **NOT** to expire the stream file and its associated volumes. The expiry date is passed directly to MVS allocation processing and is not used by ExHPDM in any of its own expiry processing. ExHPDM expiration is performed using the client JCL **EXPDT**, **RETPD** or **MANAGEMENT** specifications and the STREAM **RETentionPerioD** specifications. See "STREAM Keyword" on page 142 for further information.

Using ExHPDM in a Disaster Recovery Process

In a disaster recovery, it is recommended that general system recovery be completed before using ExHPDM to recover data. Performance optimization features such as multiple client streaming and 256KB blocks mean that only ExHPDM can read ExHPDM-managed tapes. Should the ExHPDM database be unavailable, ExHPDM can be run in Disaster Recovery mode to recover data from backups. See "ExHPDM in Disaster Recovery Mode" on page 255.

Backing Up ExHPDM Database for Disaster Recovery

For the purpose of this description, a disaster is a situation that prevents access to a substantial quantity of data stored in ExHPDM stream files. The main possible causes of such a situation are:

- Catastrophic loss where system functions other than just ExHPDM are unavailable
- Loss of the ExHPDM database.

The first cause is beyond the scope of this description. This description addresses the problem of ensuring that an up to date ExHPDM database is available, which, in turn, allows access to all data stored in ExHPDM stream files.

The ExHPDM database is a critical resource for successful disaster recovery. If a backup of the database is not available during disaster recovery, the disaster recovery process is severely impacted. Therefore, it is important for the ExHPDM database to be correctly backed up. This can be done in the following ways:

- Using the ADMIN DATABASE **BACKUP** utility is the most preferred technique. This technique copies the complete current contents of the database to a sequential data set that can then be processed using any archival procedure.
- Using a physical full volume backup.

Note: The backup produced for the database should be external to ExHPDM. If the ExHPDM interface becomes unusable, and the database backup is stored in an ExHPDM

stream file, and the database was subsequently lost, the database would only be recoverable using the ADMIN SCANSTReamfile utility.

Checklist for Backing Up the ExHPDM Database

The following checklist should be followed when backing up the ExHPDM database. Following this procedure will maximize the probability of successful disaster recovery.

- Ensure the JCL for creating an empty ExHPDM database is available. Because this is only a few lines of code, a hardcopy of the JCL should be attached to the disaster recovery plan.
- Specify database journaling. The journal data sets should be allocated on DASD that does not have a common point of failure with the database. If cross–system sharing of the database is being used, ensure that the journals from all ExHPDM systems are accessible from the ExHPDM system that will perform the backups. This is necessary for the complete backup of the journal data set contents.
- Ensure that the ExHPDM database is backed up at regular intervals using the ADMIN DATABASE **BACKUP** command. This creates sequential database and journal backup files. The **BACKUP** command is described in detail in “Database BACKUP Processing” on page 249.
- Archive the database and journal backups and store them offsite, if possible. If ExHPDM is used to archive the backups (not recommended), ensure that a hardcopy of the job log is stored with the disaster recovery plan. This is because the correct list of volume serial numbers (of the stream file that contains the backups) must be provided to the ADMIN SCANSTReamfile command, should the command ever be required.
- Whether or not the steps above are taken, you should perform the ADMIN DATABASE **LIST** command at regular intervals and attach the output to the disaster recovery plan. The ADMIN DATABASE **LIST** command lists the most recent generations of all available data. This information is valuable should the ADMIN SCANSTReamfile command ever be required.
- Backup the startup parameter file

Restoring the ExHPDM Database in Disaster Recovery Situations

There are three steps to get ExHPDM re–established in a disaster recover situation:

1. Access the most recently archived full volume backup of the ExHPDM database.
2. Restore the ExHPDM database using the same full volume disaster recovery utility (for example, FDR or DFSMSdss) that was used to dump the backup of the database
3. Apply existing database backups and journals to bring the most recently archived database up to date.

After the ExHPDM database is restored, ExHPDM can be re–started.

Reinstating a Corrupted ExHPDM Database

The following procedures describe how to reinstate a VSAM database file that was corrupted.

The ADMIN DataBase RESTORE command restores the database from a previous backup. Before submitting this job, the ExHPDM system(s) that are sharing the database should be prepared for the restore with the following conditions:

1. The database and journal startup parameters for the ExHPDM system that is to be used for the restore process should be setup in the same way that it was for backup. This allows ExHPDM to locate the correct backups and journals.
2. The database and journal backup files should reside on a DASD that is accessible from the system that performs the restore.
3. The current database must be empty. The following procedures can be used to ensure the database remains empty:
   - If ExHPDM is running, and a restore becomes necessary, issue a SET **DB QUIESCE** command. When the system is quiesced, submit an IDCAMS delete/redefine for the database (see SAMPLIB member DEFDBASE).

     or

   - If ExHPDM is shutdown, submit the IDCAMS delete/redefine for the database, then either:
     - edit the startup parameter file to add the **QUIESCE** parameter and start ExHPDM, or
     - start ExHPDM using the **QUIESCE** start parameter.
4. All ExHPDM systems must have their database quiesced, including the system that performs the restore. Preferably, the other systems should not even be started. The system that performs the restore must be up and running.

After a restore operation is successfully completed, the SET **DB RESTART** command should be issued to all applicable ExHPDM systems.

Hint: You should use the same system to perform the restore that performed the backup, unless it is impossible to do so. In addition, you should either use journaling or not use journaling consistently. Switching between journaling and not journaling during normal ExHPDM operation can cause missing data.

For detailed information on restoring an ExHPDM database please refer to “Database RESTORE Processing” on page 248.

Restoring from a REPROed Backup

The following procedure can be used if it is absolutely required to restore the ExHPDM database from a backup that was created by IDCAMS repro rather then the recommended technique described in “Database BACKUP Processing” on page 249.

Warning: You should not use the VSAM IDCAMS utility to REPRO the database into a sequential data set. A database restored from this copy would normally be unsuitable for continued use by ExHPDM.

1. Quiesce the database on all ExHPDMs sharing the database or shut down all ExHPDM servers.
2. Using the original ExHPDM database definition, change the share options to (1,3) and delete redefine.
3. Use IDCAMS REPRO to copy the backup database to the newly defined database.
4. Restart the database on one (1) ExHPDM server only.
5. Issue ExHPDM ADMIN DataBase **BACKUP**. This creates a usable backup.
6. Quiesce the database again.
7. Submit IDCAMS DELETE/DEFINE with the normally used definitions and then issue the ADMIN DataBase **RESTORE** command.
8. When successful, restart the database to all reacquired ExHPDM servers.

Recovering from a Full Journal Data Set

If the MONITOR keyword of the startup parameter file is specified and the journal data sets show a high percentage of used space or when the journal data sets fill, the following procedure can be used to correct the condition.

1. All activity for ExHPDM should be stopped. This may happen automatically if the ExHPDM database is quiesced because a journal is full. If the database is not quiesced, issue the SET **DATABASE QUIESCE** operator command to quiesce the database on all applicable systems except the system that will perform the backup.
2. Issue the ADMIN DATABASE BACKUP command to backup the ExHPDM database.
3. Issue the SET **DATABASE QUIESCE** operator command to quiesce the database on the system that performed the backup.
4. The journal should be deleted and reallocated in a larger extent (see sample library member SOVDBDEF).
5. Issue the SET **DATABASE RESTART** operator command to restart the database on all applicable ExHPDM systems.
6. ExHPDM processing is ready to resume.

Database Recovery If the Database Cannot Be Started

If the database cannot be started or restarted, it is possible to fix the database without losing any data in the database.

If ExHPDM cannot open the database because it only has one CA; DFSMShsm has migrated and recalled or was used to recover the database; or some other VSAM utility was used on the database, and stringent share options are in effect, perform the following:

1. Use IDCAMS ALTER to change the share options to (1,3). This may be the quickest way to get ExHPDM going in an emergency, although sharing of the database is then prohibited.
2. Restart the database on one (1) ExHPDM server only.
3. Issue ExHPDM ADMIN DataBase **BACKUP**. This creates a usable backup.
4. Quiesce the database again.
5. Submit IDCAMS DELETE/DEFINE with the normally used definitions and then issue the ADMIN DataBase **RESTORE** command.
6. When successful, restart the database to all reacquired ExHPDM servers.

The Consumption of Address Space IDs (ASID) in MVS

ExHPDM makes extensive use of space switch MVS cross memory services (PC–SS) to provide its interaddress space connections. To use the MVS cross memory services, ExHPDM requires the utilization of a system linkage index that is associated to that address space number (ASID) for the life of the IPL.

It is for this reason, that after termination of an instance of the ExHPDM server, it's address space becomes non–reusable in that IPL. This is symptomatic of any application that uses MVS system wide cross memory services.

The following message is issued by MVS to indicate this condition when the ExHPDM server address space terminates:

```
IEF352I ADDRESS SPACE UNAVAILABLE
```

The issuance of this message does not prevent ExHPDM from being stopped and restarted. It simple means that ExHPDM uses another ASID next time it is started in the same IPL.

ExHPDM Diagnostic Facilities

ExHPDM uses the following diagnostic facilities that are provided in the MVS operating system and in ExHPDM:

- ExHPDM VERBOSE messages
- MVS dump services
- ExHPDM Internal Tracing Facility (ITF)
- MVS GTF tracing services
- MVS system trace
- MVS master trace as recorded in the SYSLOG data set.

ExHPDM VERBOSE Messages

ExHPDM generates additional diagnostic messages to its log file when VERBOSE messages are requested. These messages are generated using message numbers starting from SOV90000. These messages are not documented in the *ExHPDM Messages and Codes* manual. They are used to assist with ExHPDM problem determination. In general, VERBOSE should not be specified unless requested to do so by StorageTek Software Support.

ExHPDM VERBOSE messages are activated by one of the following:

- ExHPDM SET VERBOSE command "SET Command" on page 200.
- ExHPDM VERBOSE parameter as described by "Chapter 7. Starting the ExHPDM Server" on page 175
- ExHPDM VERBOSE parameter as described by "MVS Subsystem Definition (IEFSSNxx)" on page 283.

ExHPDM VERBOSE messages are only output when log file messages are being directed to a data set or to SYSOUT. No VERBOSE messages are output when the LOGFILE WTO option has been specified.

MVS Dump Services

ExHPDM uses the dump services of the MVS operating system to provide a snapshot of the error or abnormal circumstance, the environment of ExHPDM, the MVS data areas, the ExHPDM internal trace facility (ITF), and other execution time resources to assist with ExHPDM problem determination.

Where possible, ExHPDM code captures programmatic errors and reflects them as external dumps. Such dumps are recorded as either:

- MVS SVC dumps to the SYSUDUMP file, or
- MVS SVC dumps to the MVS system dump files when the SYSUDUMP file has not been specified at ExHPDM startup.

ExHPDM Internal Tracing Facility (ITF)

ExHPDM builds and maintains an internal trace resource that is continuously active to record ExHPDM internal events.

The Internal Tracing Facility is not external and is used by ExHPDM technical support to trace the most current ExHPDM events that have taken place within ExHPDM.

Internal Tracing Facility events comprise of diagnostic information that may be used to resolve problems. The Internal Tracing Facility buffer is always available for an address space dump of the ExHPDM server address space.

The Internal Tracing Facility is a circular trace buffer that overwrites (wraps) older events with newer events. The events recorded by the Internal Tracing Facility may be captured using the MVS GTF tracing service.

MVS GTF Tracing Services

ExHPDM uses the MVS Generalized Trace Facility (GTF) to externally record ExHPDM trace events as recording by the ExHPDM Internal Tracing Facility.

GTF tracing of ExHPDM may be requested by ExHPDM technical support. To perform tracing GTF must be active for TYPE=USR or TYPE=USRP records and ExHPDM tracing active. ExHPDM GTF tracing is activated by one of the following:

- ExHPDM SET TRACE command “SET Command” on page 200.
- ExHPDM TRACE parameter as described by “Chapter 7. Starting the ExHPDM Server” on page 175
- ExHPDM TRACE parameter as described by “MVS Subsystem Definition (IEFSSNxx)” on page 283.

GTF recording of the ExHPDM trace records provides the only way in which to collect or record all internal tracing events over a period of time. The record of such events is used by ExHPDM technical support to diagnose problems.

ExHPDM trace records are formatted using the GTF formatting utilities available using IPCS in MVS.

MVS System Trace

The MVS system trace is maintained by the MVS operating system to record general events for all MVS address spaces and operating system component software.

The MVS trace is a circular table that is captured with an MVS dump of the ExHPDM address space. This trace is formatted using utilities available in the MVS operating system.

The MVS trace is used by ExHPDM technical support in conjunction with other information supplied by the user to diagnose ExHPDM problems.

MVS Master Trace

The MVS master trace provides a journal of all MVS console traffic on the MVS system. The master trace is traditionally referred to as SYSLOG. •The MVS master trace is often maintained by the primary Job Entry Subsystem (JES) on an MVS system.

The MVS master trace is used by both the user and ExHPDM technical support to review events leading up to and including the problem to be identified and diagnosed.

ExHPDM and MVS Time Changes

ExHPDM's use of the MVS TOD Clock

When a new client attaches to the ExHPDM server, it uses the TOD clock value at connect time as the client creation date and saves this value in the ExHPDM database. When a database list function is performed, ExHPDM converts this TOD clock value to the

displayed value by adjusting it with the timezone offset stored in the CVT. This means that if you are using a copy of your ExHPDM database at a disaster recovery site, then to ensure that a database list operation shows the same date as at your normal site, you must ensure the offset stored in the CVT is the same between the two sites.

MVS Time Changes

ExHPDM must be stopped and restarted when there is any change in the MVS local time, such as when daylight savings time begins or ends. If the MVS image, where ExHPDM runs, is IPLed as part of the change in MVS local time, then this is sufficient. If however, the MVS image is not IPLed and, for example, MVS commands are used to change the MVS local time, then ExHPDM will have to be stopped and restarted to set the time change.

This is due to the fact that ExHPDM uses the SAS/C time functions. When used the first time, the SAS/C time functions save the GMT offset as specified in the CLOCKnn member of SYS1.PARMLIB and as found in the MVS CVT control block. It uses the saved value in all future time function calls. If the GMT offset is changed in the CVT, for example by the MVS SET CLOCK command, SAS/C does not recognize this change and ExHPDM will continue to display the old time. To fix this problem, ExHPDM must be stopped before the time change is made and restarted again after the time change has been completed.

Chapter 4. Optimizing ExHPDM Performance

Overview

ExHPDM is dependent on the installer's input for optimal performance. While ExHPDM performs exceptionally as loaded, additional parameters can be specified to increase performance.

The parameters that control the optimization of ExHPDM performance are in the ExHPDM startup parameter file

The items that have the greatest impact on ExHPDM's performance are:

- the available tape devices, how many, their speed, and data rate
- the amount of data activity
- the I/O paths that are available
- the backup and restore systems already in place.

Additional information that is used by ExHPDM contains:

- the size of the jobs that make up the job mix
- the number of jobs that run during the backup window
- the length of time allotted for the backup window.

When the job mix, the number of jobs run, the time, and the tape device data rate have been determined, this information is used to calculate the approximate values to be supplied to the ExHPDM server for load balancing and scheduling ExHPDM activity.

To get an approximate value for the flow limit of a stream; add up the data rates of the tape devices and divide by the number of tape devices. The answer is the tasks in parallel to the tape devices.

To determine the concurrent number of client connects per stream; divide the device flow rate by the expected flow increase and add one.

Optimizing the performance of ExHPDM involves two methods:

- load balancing
- scheduling.

Load Balancing

Load balancing during write processing is provided in ExHPDM by the following keywords in the ExHPDM startup parameter file; STREAM, CLASS, and DEVICE. The parameters of these keywords that control write load balancing are:

CONNECTIONS
Limit of connections to an occurrence of a STREAM, that is, a stream task. CONNECTIONS is specified on the STREAM keyword.

ExpectedFlowIncrease
This is the expected performance that a single connection using the CLASS will have. ExpectedFlowIncrease is specified on the CLASS keyword.

FLOWLIMIT
Target limit performance ceiling that is applied for connections that have the same DEVICE for an occurrence of a STREAM, that is, a stream task. FLOWLIMIT is specified on the DEVICE keyword.

Note: All load balancing parameters are ignored during read processing.

Figure 4. on page 63 shows the relationship of these STREAM, CLASS, and DEVICE keywords to the connections that may use a STREAM and how the parameters for these keywords are used in load balancing.

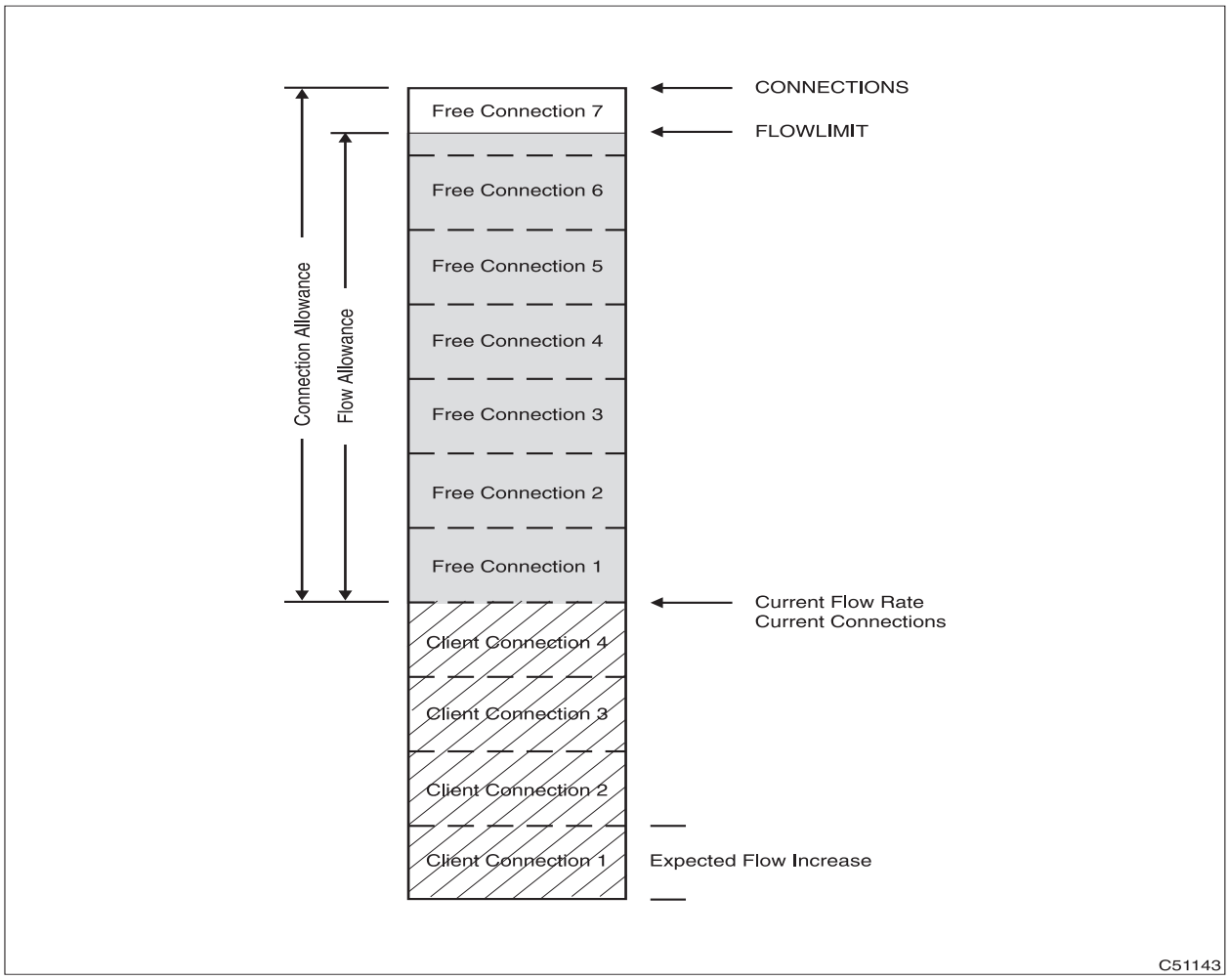


Figure 4. Stream Load Balancing Using the CLASS Limits

The **CONNECTIONS** parameter of the STREAM keyword and the **FLOWLIMIT** parameter of the DEVICE keyword denote the limits of client parallel connections to a stream task.

CONNECTIONS indicates an absolute limit. Once the CONNECTIONS limit is reached for a stream task, no new connections are allowed to that stream task until an existing connection completes. Current connections never exceed CONNECTIONS.

FLOWLIMIT is a target limit. No new connections are allowed to a specific stream task when the FLOWLIMIT is exceeded. A new client connection is always allowed to a stream task where the FLOWLIMIT has not been reached. Current flow rate can exceed FLOWLIMIT.

Each client connection and the ExpectedFlowIncrease parameter of the CLASS keyword specify the increments applied to reach these limits. The current flow rate is the summation of ExpectedFlowIncrease for all the currently connected clients to the stream

task. The current connections are the number of clients currently connected to the stream task. Where ExpectedFlowIncrease is zero or not specified, it does not increase the current flow rate.

The amount of flow left in a stream task is calculated as **FLOWLIMIT** minus current flow rate. This is the flow allowance.

The number of connections left in a stream task is calculated as **CONNECTIONS** minus current connections. This is the connection allowance.

When the CONNECTIONS parameter limit is reached (connection allowance is zero) or the FLOWLIMIT parameter is exceeded (flow allowance is less than zero) for all current stream tasks, then new connections result in another stream task being started up to the limits imposed by the CONCURRENT parameter of the STREAM keyword or the DEVICELIMIT parameter of the DEVICE keyword. If either of these limits is reached, then these new connections are processed as follows:

- If the STREAM CONCURRENT limit is reached and there are other CLASS STREAMs listed in the new client connections, then these other streams are given the connection up to their CONCURRENT limit,
- If the DEVICE DEVICELIMIT is reached and there are other STREAM DEVICEs listed in the stream definition, then these other device definitions are used up to their DEVICELIMIT,
- Otherwise, new connections either wait (WAIT(YES)) until a stream task becomes available or are rejected (WAIT(NO)). The WAIT parameter is described by the “STREAM Keyword” on page 142 and “Both BACKUP and RESTORE steps may require some time to complete. A 300-cylinder database may take between one and 20 minutes to restore, depending on system speed and other workloads. Even starting ExHPDM with a new (empty) database may take a few minutes, since the database needs to be formatted before use.” on page 272.

Examples

Example 1

This example shows the use of CLASS EFI, DEVICE FLOWLIMIT and STREAM CONNECTIONS in balancing the ExHPDM work load. In Figure 5. on page 65, the stream definition could appear as:

```
STREAM (STREMEX1 DSN(STREMEX1.SRVR1)
CONNECTIONS(11) CONCURRENT(5) DEV(SD3))
```

In Figure 5., the class definition could appear as:

```
CLASS (CLASSEX1 EFI(1M) STREAM(STREMEX1))
```

In Figure 5., the device definition could appear as:

```
DEVICE (SD3 FLOWLIMIT(10M) UNIT(SD3ESOT) DEVL(4))
```

When the CONNECTIONS parameter limit is reached or the FLOWLIMIT parameter is exceeded for these definitions for all stream tasks STREMEX1, another stream task is started up to the limits imposed by the CONCURRENT parameter of the STREAM keyword. Up to five concurrent stream tasks may be active in this example.

Figure 5. shows multiple occurrences of the stream task defined by a stream definition named STREMEX1. All of the connections are using the class definition CLASSEX1 and device definition SD3.

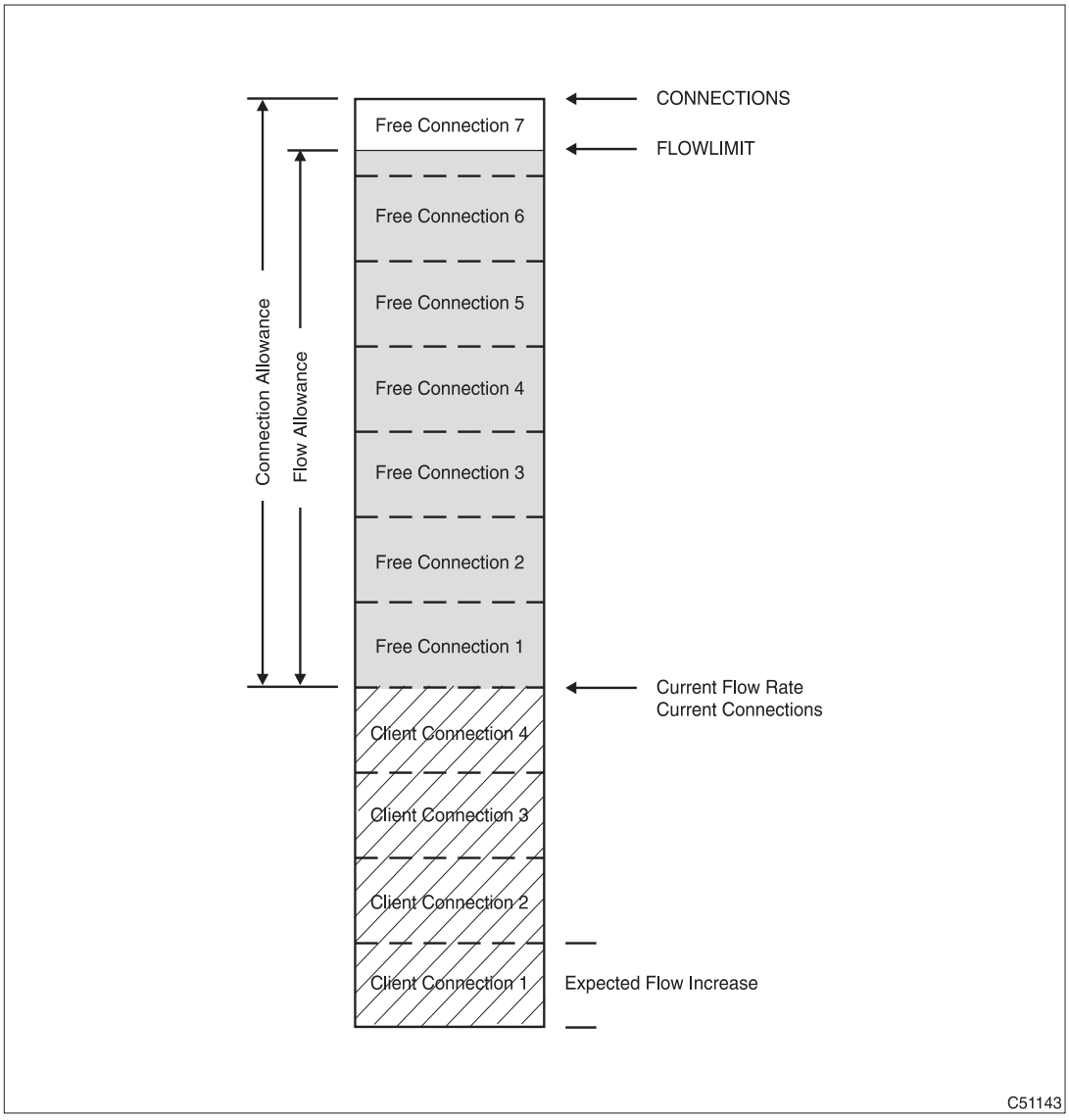


Figure 5. Load Balancing (Example 1)

Figure 5. shows that client jobs 1 through 7 make up 11 connections to the stream task STREAM named STREMEX1. This connection limit is imposed by CONNECTIONS parameter for the stream STREMEX1 and the FLOWLIMIT parameter for the device SD3. A further stream task STREAM named STREMEX1 was started for the connections

from client job 7 (connection 12) and client job 8 (connection 13). This stream task allows another nine connections from the CLASSEX1 class before starting another STREMEX1 stream task. When all five STREMEX1 stream tasks are started and fully used, any further connections from clients waits until there is a closed connection. If WAIT(NO) was specified on the STREAM keyword or on the client DD SUBSYS parameters, connections do not wait, but are rejected.

Example 2

This example shows the usage of DEVICE and CLASS definitions in the balancing of stream tasks. In Figure 6. on page 67 the stream definition could appear as:

```
STREAM (STREMEX2 DSN(STREMEX2.SRVR1) DEV(STKDEV1 STKDEV2))
```

In Figure 6., the class definition could appear as:

```
CLASS (CLASSEX2 EFI(3.5M) STREAM(STREMEX2))
```

In Figure 6. the device definitions could appear as:

```
DEVICE (STKDEV1 FLOWLIMIT(10M) UNIT(STK9840) DEVL(2))
DEVICE (STKDEV2 FLOWLIMIT(12M) UNIT(STKSD3) DEVL(1))
```

When the DEVICE FLOWLIMIT parameter is exceeded for these definitions for all stream tasks STREMEX2, another stream task is started up to the limits imposed by the number of devices available in STKDEV1 and STKDEV2. There is no connections limit specified for the STREAM STREMEX2; therefore, the default value of 10 is used. This number of connections is never reached as the CLASS EFI and DEVICE FLOWLIMIT does not allow this number of connections to a STREMEX2 stream task.

Only three concurrent stream tasks of the name STREMEX2 are allowed. This is defined by the limit of the number of devices available for STREMEX2. STKDEV1 allows for up to two devices to be used as defined by DEVICE DEVL(2). STKDEV2 allows for only a single device as defined by DEVICE DEVL(1).

Each device defined by DEVICE STKDEV1 allows up to three connections for CLASS CLASSEX2 as limited by the CLASS EFI (3.5 MB/sec) and the DEVICE FLOWLIMIT (10 MB/sec). That is, three connections are required at 3.5 MB/sec to exceed the FLOWLIMIT at 10 MB/sec. Thus, making a total of six connections for DEVICE STKDEV1.

The device defined by STKDEV2 has a higher FLOWLIMIT (12 MB/sec) than STKDEV1 and allows for up to four connections. That is, four connections are required at 3.5 MB/sec to exceed the FLOWLIMIT at 12 MB/sec.

This approach allows a total of 10 connections to be active at the same time if all devices are used (six connections for STKDEV1 and four connections for STKDEV2). Any further connections wait until the completion of an existing connection.

Figure 6. on page 67 shows multiple occurrences of the stream task defined by the stream definition named STREMEX2. All of the connections are using the class definition CLASSEX2 and devices from device definition STKDEV1 and STKDEV2.

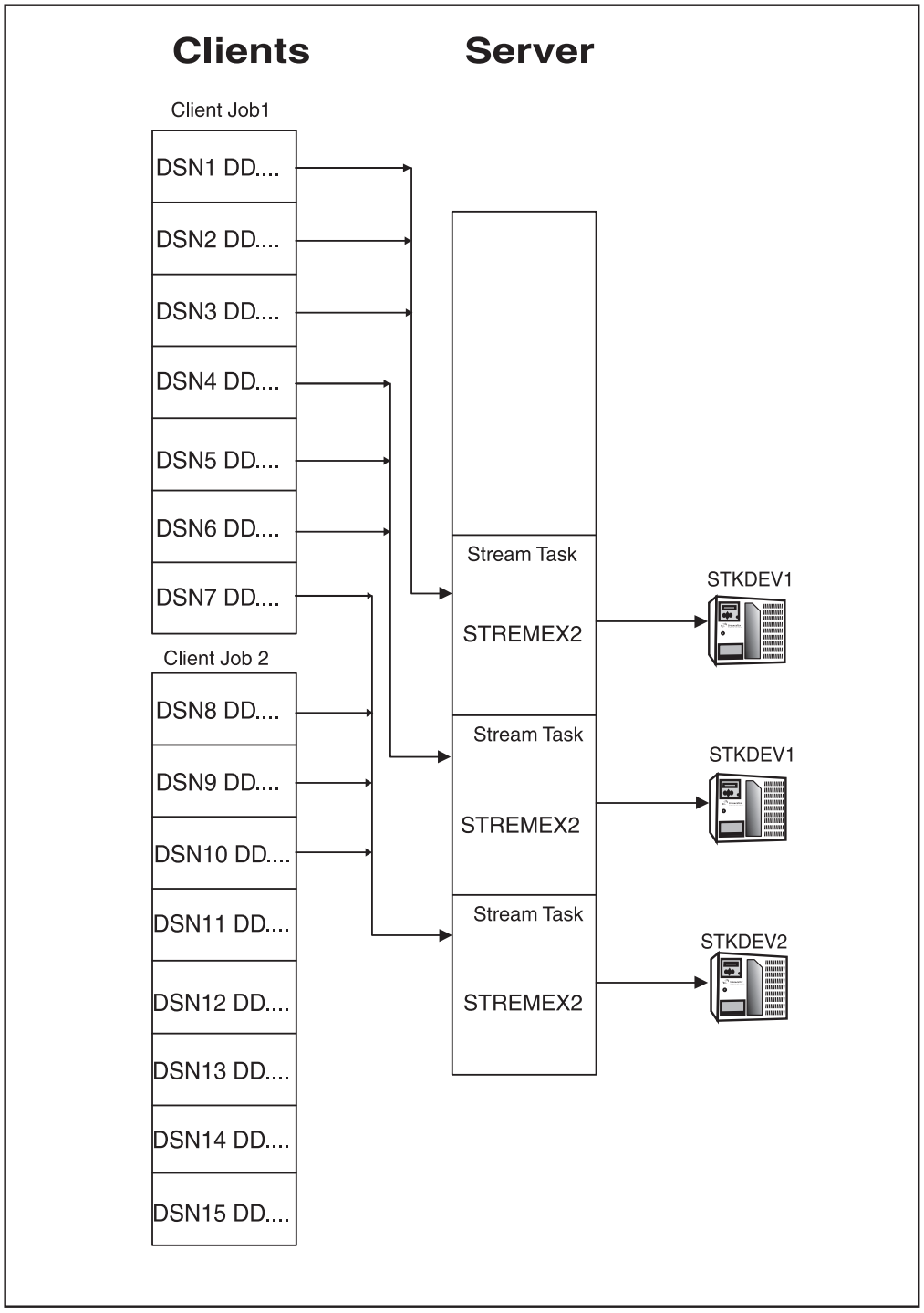


Figure 6. Load Balancing (Example 2)

In Figure 6., the client jobs make up 10 connections to the STREAM named STREMEX2. There are three stream tasks active by this name using three devices. The only current connections are for DSN1 through to DSN10. Connections for DSN11 through to DSN15

wait until an existing connection completes. This connection limit is imposed by the number of devices available for the stream STREMEX2 and the FLOWLIMIT parameter for the devices STKDEV1 and STKDEV2. If WAIT(NO) was specified on the STREAM keyword or on the client DD SUBSYS parameters, connections do not wait, but are rejected.

Example 3

The following example shows the usage of multiple stream definitions when doing performance balancing. In Figure 7. on page 70 the stream definition could appear as:

```
STREAM (STRMEX3A DSN(STRMEX3A.SRVR1)
DEV(STKDEV1) CONNECTIONS(2) CONCURRENT(WRITE(1)))
STREAM (STRMEX3B DSN(STRMEX3B.SRVR1)
DEV(STKDEV1 STKDEV2))
```

In Figure 7. on page 70 the class definition could appear as:

```
CLASS (CLASSEX3 EFI(3.5M) STREAM(STRMEX3A STRMEX3B))
```

In Figure 7. on page 70 the device definitions could appear as:

```
DEVICE (STKDEV1 FLOWLIMIT(10M) UNIT(STK9840) DEVL(2))
DEVICE (STKDEV2 FLOWLIMIT(12M) UNIT(STKSD3) DEVL(1))
```

CLASS CLASSEX3 has a choice of STREAMs STRMEX3A and STRMEX3B. STREAM STRMEX3A allows only two connections, as specified by CONNECTIONS(2), and only one stream task as specified by CONCURRENT(WRITE(1)) to be active by this name. The CLASS EFI never reaches the DEVICE FLOWLIMIT because STREAM CONNECTIONS for STRMEX3A is more limiting in this example. Once these two connections are made then further connections using the CLASS CLASSEX3 are made to STREAM STRMEX3B.

The number of connections to STRMEX3B is controlled by the CLASS EFI and the DEVICE FLOWLIMIT. When the FLOWLIMIT parameter is exceeded for these definitions for all stream tasks STRMEX3B, another stream task is started up to the limits imposed by the number of device available in STKDEV1 and STKDEV2. There is no connections limit specified by the STREAM STRMEX3A; therefore, the default value of 10 is used. This number of connections is never reached because the CLASS EFI and DEVICE FLOWLIMIT do not allow this number of connections to a STRMEX3B stream task. If the STRMEX3A stream task is currently active, then it is using one of the devices out of the DEVICE STKDEV1 definition. This approach allows STRMEX3B to use up to two devices, since the total device limits for these DEVICE definitions is three. That is, if a STRMEX3A stream task is active, then STRMEX3B can use one STKDEV1 device and one STKDEV2 device. The STKDEV1 device has a FLOWLIMIT of 10 MB/sec which allows up to three connections. That is, three connections with an EFI of 3.5 MB/sec are required to exceed the device FLOWLIMIT of 10 MB/sec. The STKDEV2 device has a FLOWLIMIT of 12 MB/sec which allows up to four connections. That is, four connections with an EFI of 3.5 MB/sec are required to exceed the DEVICE FLOWLIMIT of 10 MB/sec. So, the total connections for STREAM STRMEX3B is seven.

Since STRMEX3A allows only two connections, then the total number of connections for the example definitions are nine. Any further connections wait until the completion of an existing connection. If WAIT(NO) is specified on the STREAM keyword or on the client DD SUBSYS parameters, connections do not wait, but are rejected.

Figure 7. on page 70 shows multiple occurrences of the stream task defined by a stream definition named STRMEX3A and STRMEX3B. All of the connections are using the class definition CLASSEX3 and devices from device definition STKDEV1 and STKDEV2. Connections for DSN3 through to DSN5 and DSN13 wait for one of the other connections to complete.

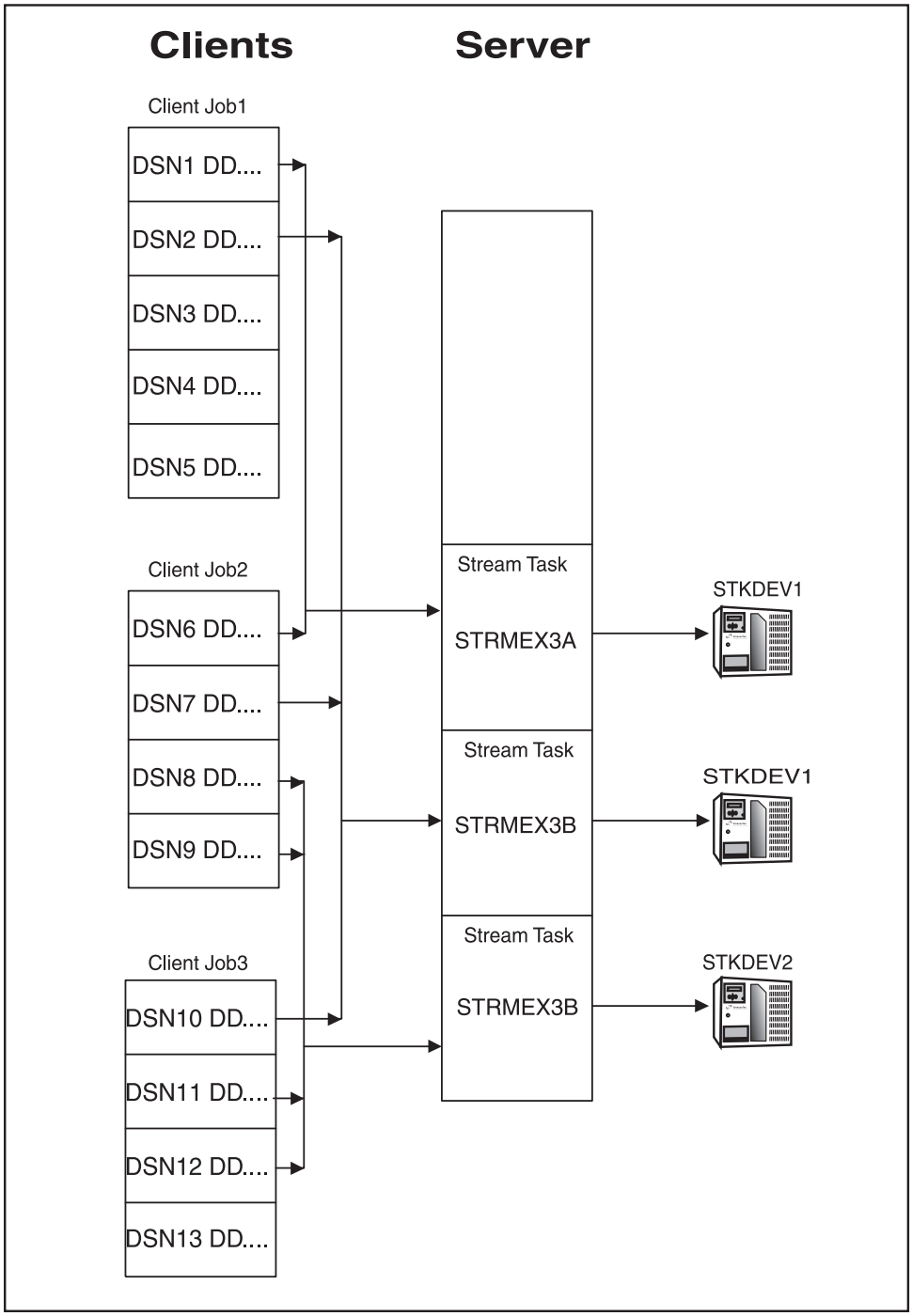


Figure 7. Load Balancing (Example 3)

Chapter 5. ExHPDM Scenarios

ExHPDM Scenarios

The following scenarios are typical examples of using ExHPDM. Additional samples are provided in the STKSAMP data set which is created during the ExHPDM installation process.

Basic ExHPDM Server Started Task JCL

The following is an example of JCL in a procedure library that is used to start an ExHPDM server called SOVREM. This JCL procedure is placed in a procedure library that is used by the primary subsystem (JES2 or JES3). An APF load library (EXHPDM.STKLOAD) is required for this procedure.

```
//SOVREM PROC
//SOVREM EXEC PGM=SOVMAIN,
//                TIME=1440,REGION=0M
//STEPLIB   DD DISP=SHR,DSN=EXHPDM.STKLOAD
//STKPARMS DD DISP=SHR,DSN=EXHPDM.STKSAMP contains SOVPRMxx members
//SYSPRINT DD SYSOUT=* required for diagnostics
//SYSTERM  DD SYSOUT=* required for diagnostics
```

STARTING the ExHPDM Server

The following is an example of starting the ExHPDM server from the MVS MCS console using an MVS START command. In this example, the standard JCL procedure SOVREM is started with an overriding specification to use parameter member SOVPRM09.

Basic ExHPDM SOVPRM*xx* Parameter File Used for Startup

The following is an example of a SOVPRM*xx* parameter file used by the ExHPDM server at startup. For this example, the name of the startup parameter file is SOVPRM09 and it is contained in the PDS named EXHPDM.STKSAMP. This is referenced in the tasked task procedure by the STKPARMS DD.

```
DATABASE(EXHPDM.T012509.SMDBASE)
   BACKUPDSN(EXHPDM.BACKUP.SAMPLE.DBASE UNIT(SYSDA))

JOURNAL(EXHPDM.&SYSNAME..&JOBNAME..JOURNAL)
   BACKUPJOURNAL(EXHPDM.BACKUP.SAMPLE.JOURNAL UNIT(SYSDA))

LOGFILE(SYSOUT(Y))
```

```
TMS(CA1)

MONITOR(DATABASE(THRESHOLD(76))
   JOURNAL(THRESHOLD(80)))

SELECT(TLRULE CLASS(TLCLASS) DSN(DUMP.TL.**))
SELECT(REDRU3A CLASS(REDCL3A) DSN(DUMP.RD3A.**))
SELECT(REDRU3E CLASS(REDCL3E) DSN(DUMP.RD3E.**))
SELECT(REDFDR CLASS(FDR1) PGM(FDR*))
SELECT(OFFSITE CLASS(OFFSITE) DSN(**.OFFSITE.**))

CLASS(REDCL3A STREAM(REDST3A) MGMT(UNCAT))
CLASS(REDCL3E STREAM(REDST3E) MGMT(UNCAT))
CLASS(TLCLASS EXPECTEDFLOWINCREASE(3M) STREAM(TLSTRM) MGMT(UNCAT))
CLASS(FDR1 EXPECTEDFLOWINCREASE(4M) STREAM(REDST3A) MGMT(KEEP10))
CLASS(OFFSITE STREAM(OFFSITE))

MGMT(UNCAT EXPIRY(WHENUNCATALOGED))
MGMT(KEEP10 EXPIRY(RETAINPERIOD(10)))

STREAM(OFFSITE DSN(HPDM.OFFSITE) DEVICE(REDDEV) WAIT(NO))
STREAM(TLSTRM DSN(HPDM.HPDMETL.STREAM) DEVICE(TLDEV)
   CONNECTIONS(20) CONCURRENT(READ(5) WRITE(4))
   WAIT(YES))
STREAM(REDST3A DSN(HPDM.HPDMD3A.STREAM)
   DEVICE(REDDEV)
   CONNECTIONS(15) CONCURRENT(READ(2) WRITE(2))
   WAIT(YES))
STREAM(REDST3E DSN(HPDM.HPDMD3E.STREAM)
   DEVICE(MAGDEV)
   CONNECTIONS(20) CONCURRENT(READ(5) WRITE(2))
   WAIT(YES) LOG(YES))

DEVICE(TLDEV UNIT(HPDMTL) FLOWLIMIT(0) DEVICELIMIT(4)
   RETAIN(10M) DEFAULT)
DEVICE(REDDEV UNIT(HPDMRW) FLOWLIMIT(12M)
   DEVICELIMIT(2)
   RETAIN(0) UNITCOUNT(2))
DEVICE(MAGDEV UNIT(HPDMMAG) FLOWLIMIT(0)
   DEVICELIMIT(2)
   RETAIN(5S))
```

This example of a parameter file is to be used with the examples that follow it.

SMC Example

This release of ExHPDM and SMC V6.2 have been changed to provide the ability for batch tape jobs to be directed to ExHPDM for processing. This is achieved via new SMC V6.2 policy statements. For example:

```
TAPEREQ JOBNAME(DFDSSJOB) POLICY(EXHPDM)
TAPEREQ JOBNAME(FDRBTJOB) POLICY(SSNAME)

POLICY NAME(EXHPDM) IDAXSUBSYS(SOV1) IDAXPROGRAM(SOVDSSU)
POLICY NAME(SSNAME) IDAXSUBSYS(SREM)
```

The policy name ExHPDM is an example of an SMC policy for a DFSMSdss job. In this example, SMC dynamically sets the DD SUBSYS parameter for the client tape dataset to be directed to ExHPDM, subsystem name SOV1. Also, the jobs program name is changed to SOVDSSU from ADRDSSU for the ExHPDM UIM to be executed. Once MVS allocation is complete, then the normal ExHPDM connection processing continues based upon the ExHPDM parameter SELECT, CLASS, STREAM parameters etc.

The policy name SSNAME is an example of an SMC policy for an FDR job. In this example, SMC dynamically sets the DD SUBSYS parameter for the client tape dataset to be directed to ExHPDM, subsystem name SREM. Once MVS allocation is complete, the FDR job tape datasets are directed to the specified EXHPDM address space. For further information, refer to the *SMC 6.2 Configuration and Administration Guide*.

DFSMSdss Examples

Basic ExHPDM Backup

Using ExHPDM and SOVDSSU is identical to using DFSMSdss and ADRDSSU. Backing up data with ExHPDM is seamless and invisible after the JCL is modified to form the connection between the JCL DDNAME statement and the ExHPDM server.

The job in the following example performs three concurrent full volume physical backups of volumes HPDM01, HPDM02, and TSO189 in parallel. The DD statements DD01, DD02 and DD03 are processed by the ExHPDM server whose subsystem name is SOV. These volumes are backup to the stream called REDST3A on the tape device known by the esoteric name HPDMRW. This is due to the processing by the SELECT statement REDRU3A. The server writes these data sets to a stream file on tape whose name is prefixed by ‘HPDM.HPDMD3A.STREAM’.

```
//DMPJOB  JOB jobcard info
//*  JOB BACKS UP 3 PACKS (IN PARALLEL)
//STEP001      EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB      DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT     DD  SYSOUT=*
//DD01         DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DD02         DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DD03         DD  UNIT=SYSDA,VOL=SER=TSO189,DISP=OLD
//OUT01        DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//                 DISP=(,CATLG,DELETE)
//OUT02        DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
```

```
//                DISP=(,CATLG,DELETE)
//OUT03    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.TSO189,
//                DISP=(,CATLG,DELETE)
//SYSIN          DD  *
 PARALLEL
 DUMP FULL INDD(DD01)  OUTDD(OUT01)
 DUMP FULL INDD(DD02)  OUTDD(OUT02)
 DUMP FULL INDYN(TSO189) OUTDD(OUT03)
```

Importantly, the DFSMSdss keyword, PARALLEL, is added to the JCL. This insures that DFSMSdss starts all the desired activities at once. This is where ExHPDM ensures that all the activities are done together through DFSMSdss.

Basic ExHPDM Full Volume Physical Restore in Parallel

The job in the following example performs two concurrent full volume physical restores of volumes HPDM01 and HPDM02 in parallel. The DD statements DD01 and DD02 are processed by the ExHPDM server whose subsystem name is SOV.

The server requests tape volumes to be mounted on tape devices whose description was specified in the stream definition named REDST3A.

```
//RESJOB  JOB jobcard info
//*  JOB RESTORES 2 PACKS (IN PARALLEL) FROM EXHPDM
//STEP001 EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB      DD DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT     DD SYSOUT=*
//OUT01        DD UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//OUT02        DD UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DD01         DD SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//                DISP=OLD
//DD02         DD SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
//                DISP=OLD
//SYSIN        DD *
 PARALLEL
 RESTORE FULL PURGE INDD(DD01) OUTDD(OUT01)
 RESTORE FULL PURGE INDD(DD02) OUTDD(OUT02)
```

Basic ExHPDM Logical Data Set DUMPS in Parallel

The job in the following example performs four logical data set dumps in parallel. The DD statements OWAS, OWCH, OWPT, and OWPL are processed by the ExHPDM server whose subsystem name is SOV.

```
//DMPJOB  JOB jobcard info
//STEP001    EXEC PGM=SOVDSSU,REGION=0M
//STEPLIB    DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT   DD SYSOUT=*
//OWAS       DD DISP=(,CATLG,DELETE),SUBSYS=SOV,
//           DSN=DUMP.RD3A.HPDMA1.COMP30PC.DMP044
//OWCH       DD DISP=(,CATLG,DELETE),SUBSYS=SOV,
//           DSN=DUMP.RD3A.HPDMA2.COMP30PC.DMP044
//OWPT       DD DISP=(,CATLG,DELETE),SUBSYS=SOV,
//           DSN=DUMP.RD3A.HPDM03.COMP30PC.DMP044
//OWPL       DD DISP=(,CATLG,DELETE),SUBSYS=SOV,
```

```
//         DSN=DUMP.RD3A.HPDM04.COMP30PC.DMP044
//SYSINDD *
 PARALLEL
 DUMP DATASET(INCLUDE(CLIENT.RD3A.HPDMA1.COMP30PC.**))  -
       OUTDDN(OWAS) TOLERATE(ENQFAILURE)
 DUMP DATASET(INCLUDE(CLIENT.RD3A.HPDMA2.COMP30PC.**))  -
       OUTDDN(OWPL) TOLERATE(ENQFAILURE)
 DUMP DATASET(INCLUDE(CLIENT.RD3A.HPDM03.COMP30PC.**))  -
       OUTDDN(OWCH) TOLERATE(ENQFAILURE)
 DUMP DATASET(INLCUDE(CLIENT.RD3A.HPDM04.COMP30PC.**))  -
       OUTDDN(OWPT) TOLERATE(ENQFAILURE)
```

Basic ExHPDM Logical Data Set RESTORES in Parallel

The job in the following example performs four logical data set restores in parallel.

```
//RESJOB  JOB jobcard info
//STEP001      EXEC PGM=SOVDSSU,REGION=0M
//STEPLIB      DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT     DD  SYSOUT=*
//HPDMA1       DD  DSN=DUMP.TL.HPDMA1.COMP30PC.DMP044,
//                 SUBSYS=SOV,DISP=OLD
//HPDMA2       DD  DSN=DUMP.TL.HPDMA2.COMP30PC.DMP044,
//                 SUBSYS=SOV,DISP=OLD
//HPDM03       DD  DSN=DUMP.TL.HPDM03.COMP30PC.DMP044,
//                 SUBSYS=SOV,DISP=OLD
//HPDM04       DD  DSN=DUMP.TL.HPDM04.COMP30PC.DMP044,
//                 SUBSYS=SOV,DISP=OLD
//SYSIN        DD  *
 PARALLEL
 RESTORE DATASET (INLCUDE(CLIENT.TL.HPDMA1.COMP30PC.**)) -
                  RENAMEU(PROD1) INDD(HPDMA1)  -
                  TOLERATE(ENQFAILURE) OUTDYN(HPDM04)
 RESTORE DATASET (INCLUDE(CLIENT.TL.HPDM02.COMP30PC.**)) -
                  RENAMEU(PROD2)  INDD(HPDMA2)  -
                  TOLERATE(ENQFAILURE) OUTDYN(HPDM04)
 RESTORE DATASET (INCLUDE(CLIENT.TL.HPDM03.COMP30PC.**)) -
                  RENAMEU(PROD3)  INDD(HPDM03)  -
                  TOLERATE(ENQFAILURE) OUTDYN(HPDM04
 RESTORE DATASET (INCLUDE(CLIENT.TL.HPDM04.COMP30PC.**)) -
                  RENAMEU(PROD04) INDD(HPDM04)  -
                  TOLERATE(ENQFAILURE) OUTDYN(HPDM04)
```

Basic ExHPDM Physical TRACK DUMPs in Parallel

The job in the following example performs three concurrent track dumps of volumes HPDM01, HPDM02, and TSO189 in parallel. the DD statements OUT01, OUT02, and OUT03 are processed by the ExHPDM server whose subsystem name is SOV.

The server requests tape volumes to be mounted on tape devices whose description was specified in the stream definition named REDST3A.

Volumes HPDM01, HPDM02, and TSO189 are allocated dynamically by SOVDSSU.

```
//DMPJOB  JOB jobcard info
//*  PERFORM SPECIFIC TRACK DUMPS (IN PARALLEL)
//STEP001    EXEC PGM=SOVDSSU,REGION=0M
```

```
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT DD  SYSOUT=*
//OUT01      DD  SUBSYS=SOV,DSN=DUMP.RD3A.TRACK.HPDM01,
//              DISP=(,CATLG,DELETE)
//OUT02       DD  SUBSYS=SOV,DSN=DUMP.RD3A.TRACK.HPDM02,
//              DISP=(,CATLG,DELETE)
//OUT03      DD  SUBSYS=SOV,DSN=DUMP.RD3A.TRACK.TSO189,
//              DISP=(,CATLG,DELETE)
//SYSIN      DD  *
 PARALLEL
 DUMP TRACKS(100,14,312,14) INDY(HPDM01)  -
              OUTDDNAME(OUT01) OPTIMIZE(4) WAIT(2,2)
 DUMP INDYNAM(HPDM02) TRACKS(2010,0,3300,14)  -
              OUTDDNAME(OUT02)
 DUMP TRACKS(005,01,773,14) INDY(TSO189)  -
              OUTDDNAME(OUT03) OPTIMIZE(4) WAIT(2,2)
```

Basic ExHPDM Physical TRACK RESTORES in Parallel

The job in the following example performs three concurrent track restores of volumes HPDM01, HPDM02 and TSO189 in parallel. The DD statements IN01, IN02, and IN03 are processed by the ExHPDM server whose subsystem name is SOV.

The server requests tape volumes to be mounted on tape devices whose description was specified in the stream definition called REDST3A

Volumes HPDM01, and TSO189 are allocated dynamically by SOVDSSU.

```
//RESJOB  JOB jobcard info
//*  PERFORM SPECIFIC TRACK RESTORES IN PARALLEL.
//STEP001      EXEC PGM=SOVDSSU,REGION=0M
//STEPLIB      DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT    DD  SYSOUT=*
//IN01    DD  SUBSYS=SOV,DSN=DUMP.RD3A.TRACK.HPDM01,
//              DISP=OLD
//IN02          DD  SUBSYS=SOV,DSN=DUMP.RD3A.TRACK.HPDM02,
//              DISP=OLD
//IN03          DD  SUBSYS=SOV,DSN=DUMP.RD3A.TRACK.TSO189,
//              DISP=OLD
//OUT02        DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//SYSIN         DD  *
 PARALLEL
 RESTORE TRACKS(100,14,312,14) INDDNAME(IN01)  -
              OUTDYN(HPDM01) OPTIMIZE(4) WAIT(2,2)
 RESTORE INDDNAME(IN02) TRACKS(2010,0,3300,14)  -
              OUTDDNAME(OUT02)
 RESTORE TRACKS(005,01,773,14) INDDNAME(IN03)  -
              OUTDYN(TSO189) OPTIMIZE(4) WAIT(2,2)
```

Performing Dump Duplexing with ExHPDM

The following example demonstrates how to perform DFSMSdss duplexing with ExHPDM. The client data sets specified by DDnames DUPLX02 and DUPLX03 are processed by stream class REDCL3E. This causes a copy of the client dump data sets to be written to DDnames DUPLX02 and DUPLX03 in addition to OUT02 and OUT03. Note that no additional copy is taken for the dump to OUT01.

The **'CLASS=REDCL3E'** specification in the JCL **SUBSYS** parameter directs the DDnames DUPLX02 and DUPLX03 to the stream called REDCL3E. This causes a new stream task to be started with a separate and new tape drive and volume. This use of a SELECT**(REDRU3E)** CLASS(**REDCL3E)** rule can also be employed to do this as well.

```
//DMPJOB  JOB jobcard info
//*  JOB BACKS UP 3 PACKS AND DUPLEXES TWO STREAM FILES.
//STEP001  EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT DD  SYSOUT=*
//DD01     DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DD02     DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//OUT01    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//             DISP=(,CATLG,DELETE)
//OUT02    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02,
//             DISP=(,CATLG,DELETE)
//OUT03    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.TSO189,
//             DISP=(,CATLG,DELETE)
//DUPLX02  DD  SUBSYS=(SOV,'CLASS(REDCL3E)'),
//             DSN=DUMP.RD3A.FULL.HPDM02,DUPLEX,
//             DISP=(,CATLG,DELETE)
//DUPLX03  DD  SUBSYS=(SOV,'CLASS(REDCL3E)'),
//             DSN=DUMP.RD3A.FULL.TSO189,DUPLEX,
//             DISP=(,CATLG,DELETE)
//SYSIN    DD  *
 PARALLEL
 DUMP FULL INDD(DD01) OUTDD(OUT01)
 DUMP FULL INDD(DD02) OUTDD(OUT02,DUPLX02)
 DUMP FULL INDYN(TSO189) OUTDD(OUT03,DUPLX03)
```

Basic ExHPDM COPYDUMP in Parallel

The following example demonstrates how to perform DFSMSdss COPYDUMP with ExHPDM. Refer to "Basic ExHPDM Backup" on page 73 for the original dump.

The use of the SELECT(OFFSITE) CLASS(OFFSITE) rule is being used to direct the copied data to the OFFSITE stream. WAIT(NO) is specified on the INDDNAMEs to avoid possible deadlocks when the input stream device is unavailable. WAIT(NO) is specified on the OFFSITE stream for the same reason. CATCHUP(ALWAYS) is specified on the input data sets to force all clients to be read on the same pass of the stream file. If this was not specified then it is possible that one or two of the clients would have to wait if their first block was already passed by the time they connect to the stream task.

```
//DMPCPY  JOB jobcard info
//*  JOB COPYDUMPs 3 PACKS (IN PARALLEL).
//STEP001      EXEC  PGM=SOVDSSU,REGION=4M
//STEPLIB      DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT     DD  SYSOUT=*
//DD01         DD  SUBSYS=(SOV,'WAIT(NO) CATCHUP(ALWAYS)'),
//        DSN=DUMP.RD3A.FULL.HPDM01
//DD02         DD  SUBSYS=(SOV,'WAIT(NO) CATCHUP(ALWAYS)'),
//        DSN=DUMP.RD3A.FULL.HPDM02
//DD03         DD  SUBSYS=(SOV,'WAIT(NO) CATCHUP(ALWAYS)'),
//        DSN=DUMP.RD3A.FULL.TSO189
//CPY01        DD  SUBSYS=SOV,
//             DSN=DUMP.RD3A.OFFSITE.HPDM01(+1),
```

```
//                DISP=(,CATLG,DELETE)
//CPY02           DD  SUBSYS=SOV,
//                DSN=DUMP.RD3A.OFFSITE.HPDM02(+1),
//                DISP=(,CATLG,DELETE)
//CPY03           DD  SUBSYS=SOV,
//                DSN=DUMP.RD3A.OFFSITE.TSO189(+1),
//                DISP=(,CATLG,DELETE)
//SYSIN           DD  *
 PARALLEL
 COPYDUMP  INDD(DD01) OUTDD(CPY01)
 COPYDUMP  INDD(DD02) OUTDD(CPY02)
 COPYDUMP  INDD(DD03) OUTDD(CPY03)
```

Using Multiple ExHPDM Servers from a Single Job

The following example demonstrates how to use multiple ExHPDM servers from a single batch job. The ExHPDM servers must be started and available in the current MVS system. The ExHPDM servers in this example are known through their MVS subsystem names of SOV and SRE9.

In this example, three volumes are dumped. Volumes HPDM01 and TSO189 are processed by the ExHPDM server available through the SOV subsystem name. Volume HPDM02 is processed by the ExHPDM server available through the SRE9 subsystem name. Note the additional specification for the client file requests directs it to a STREAM CLASS named TLCLASS and that it need not wait around (WAIT(NO)) for the ExHPDM resources if they are not available.

```
//DMPCPY  JOB jobcard info
//* Backup using different servers to spread the load.
//STEP001  EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT DD  SYSOUT=*
//DD01     DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DD02     DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//OUT01    DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01,
//             DISP=(,CATLG,DELETE)
//OUT02    DD  SUBSYS=SRE9,DSN=DUMP.RD3A.FULL.HPDM02,
//             DISP=(,CATLG,DELETE)
//OUT03    DD  DSN=DUMP.RD3A.FULL.TSO189,
//             DISP=(,CATLG,DELETE),
//             SUBSYS=(SOV,'CLASS(TLCLASS) WAIT(NO)')
//SYSIN    DD  *
 PARALLEL
 DUMP FULL INDD(DD01) OUTDD(OUT01)
 DUMP FULL INDD(DD02) OUTDD(OUT02)
 DUMP FULL INDYN(TSO189) OUTDD(OUT03)
/*
```

HFS Backup

The following example demonstrates how to perform a DFSMSdss HFS logical dataset dump, where the processing is being performed by the ExHPDM server who subsystem is SOV.

```
//LDMPJOB4 JOB jobcard info
//* Job backs up HFS data
```

```
//STEP1    EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//IN1      DD  DSN=HFS.MVS.ROOT,DISP=SHR
//TAPE1    DD  DSN=DUMP.HFS.DATA,DISP=(,CATLG,DELETE),
//         UNIT=TAPE,SUBSYS=SOV
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**)) INDD(IN1) OUTDD(TAPE1) ALLEXCP ALLDATA(*)
/*
```

Linux Backup

The following example demonstrates how to perform a DFSMSdss Linux logical partition dump, where the processing is being performed by the ExHPDM server who subsystem is SOV.

```
//LDMPJOB4 JOB jobcard info
//* Job backs up Linux partition
//STEP1    EXEC PGM=SOVDSSU,REGION=4M
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//IN1      DD  UNIT=3390,VOL=SER=LINUX1,DISP=SHR
//TAPE1    DD  DSN=DUMP.PART.DATA,DISP=(,CATLG,DELETE),
//         SUBSYS=SOV
//SYSIN    DD *
 DUMP DATASET(INCLUDE(LINUX.VLINUX1.PART0000.NATIVE)) INDD(IN1)
OUTDD(TAPE1) ALLEXCP
/*
```

FDR Examples

Basic ExHPDM Backup

Using ExHPDM and FDR is identical to using FDR without ExHPDM. Backing up data with ExHPDM is seamless and invisible after the JCL is modified to form the connection between the JCL DDNAME statement and the ExHPDM server.

The job in the following example performs three concurrent full volume physical backups of volumes HPDM01, HPDM02, and TSO189 in parallel. The DD statements TAPE1, TAPE2 and TAPE3 are processed by the ExHPDM server whose subsystem name is SOV. These volumes are backed up to the stream called REDST3A on the tape device known by the esoteric name HPDMRW. This is due to the processing by the SELECT statement REDRU3A. The server writes these data sets to a stream file on tape whose name is prefixed by 'HPDM.HPDMD3A.STREAM'.

```
//DUMPJOB    JOB jobcard info
//STEP001    EXEC  PGM=FDR,REGION=4M
//SYSPRINT   DD  SYSOUT=*
//DISK1DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DISK2DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DISK3DD  UNIT=SYSDA,VOL=SER=TSO189,DISP=OLD
//TAPE1      DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01(+1),
//           DISP=(,CATLG,DELETE)
//TAPE2      DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02(+1),
//           DISP=(,CATLG,DELETE)
//TAPE3      DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.TSO189(+1),
```

```
//                DISP=(,CATLG,DELETE)
//SYSPRIN1    DD  SYSOUT=*
//SYSPRIN2    DD  SYSOUT=*
//SYSPRIN3    DD  SYSOUT=*
//SYSIN       DD  *
 DUMP TYPE=FDR,ATTACH,MAXERR=1
/*
```

Importantly, the FDR keyword, ATTACH, is added to the JCL. This insures that FDR starts all the desired activities at once. This is where ExHPDM ensures that all the activities are processed together through FDR. Alternatively, the parameter PARM='A' could have been specified. This indicates that a DUMP is required and each of these should be attached.

```
//DUMPJOB     JOB jobcard info
//STEP001     EXEC  PGM=FDR,PARM='A',REGION=4M
//SYSPRINT    DD  SYSOUT=*
//DISK1       DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DISK2       DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//DISK3       DD  UNIT=SYSDA,VOL=SER=TSO189,DISP=OLD
//TAPE1       DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01(+1),
//              DISP=(,CATLG,DELETE)
//TAPE2       DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02(+1),
//              DISP=(,CATLG,DELETE)
//TAPE3       DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.TSO189(+1),
//              DISP=(,CATLG,DELETE)
//SYSPRIN1    DD  SYSOUT=*
//SYSPRIN2    DD  SYSOUT=*
//SYSPRIN3    DD  SYSOUT=*
```

Basic ExHPDM Full Volume Restore in Parallel

The job in the following example performs two concurrent full volume restores of volumes HPDM01 and HPDM02 in parallel. The DD statements DD01 and DD02 are processed by the ExHPDM server whose subsystem name is SOV. The most recent generations of the backups of these volumes is accessed using the GDG (0) specification.

The server requests tape volumes to be mounted on tape devices whose description was specified in the stream definition named REDST3A.

```
//RESJOB      JOB jobcard info
//STEP001     EXEC  PGM=FDR,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//DISK1DD  UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DISK2 DD  UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//TAPE1       DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM01(0)
//TAPE2       DD  SUBSYS=SOV,DSN=DUMP.RD3A.FULL.HPDM02(0)
//SYSPRIN1    DD  SYSOUT=*
//SYSPRIN2    DD  SYSOUT=*
//SYSIN       DD  *
 RESTORE TYPE=FDR,MAXTASKS=2,MAXERR=1
/*
```

Basic ExHPDM Data Set DUMPS in Parallel

The job in the following example performs four data set dumps in parallel. The DD statements TAPE1 through to TAPE4 are processed by the ExHPDM server whose subsystem name is SOV. The selection of the connections is allowed by the SELECT rule REDFDR due to the matching ProGraM(FDR*) name FDRDSF.

```
//DUMPJOB   JOB jobcard info
//STEP001     EXEC  PGM=FDRDSF,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//DISK1       DD  UNIT=SYSDA,VOL=SER=TSTVL1,DISP=OLD
//DISK2       DD  UNIT=SYSDA,VOL=SER=TSTVL2,DISP=OLD
//DISK3       DD  UNIT=SYSDA,VOL=SER=TSTVL3,DISP=OLD
//TAPE1       DD  SUBSYS=SOV,DSN=DUMP.FDRDSF.TSTVL1(+1),
//               DISP=(,CATLG,DELETE)
//TAPE2       DD  SUBSYS=SOV,DSN=DUMP.FDRDSF.TSTVL2(+1),
//               DISP=(,CATLG,DELETE)
//TAPE3       DD  SUBSYS=SOV,DSN=DUMP.FDRDSF.TSTVL3(+1),
//               DISP=(,CATLG,DELETE)
//SYSPRIN1    DD  SYSOUT=*
//SYSPRIN2    DD  SYSOUT=*
//SYSPRIN3    DD  SYSOUT=*
//SYSIN       DD  *
 DUMP TYPE=DSF,DSNENQ=USE,ATTACH,MAXERR=1
 SELECT DSN=CLIENT.RD3A.HPDMA1.COMP30PC.**
 SELECT DSN=CLIENT.RD3A.HPDMA2.COMP30PC.**
 SELECT DSN=CLIENT.RD3A.HPDM03.COMP30PC.**
 SELECT DSN=CLIENT.RD3A.HPDM04.COMP30PC.**
/*
```

Basic ExHPDM Data Set RESTORES in Parallel

The job in the following example performs three data set restores in parallel using three jobs.

Caution: FDRDSF parallel restores should not be performed if any of the restoring data sets are multivolume. A parallel restore can result in multivolume data sets being incorrectly restored.

```
//RESJOB1     JOB jobcard info
//STEP001     EXEC  PGM=FDRDSF,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//DISK1       DD  UNIT=SYSDA,VOL=SER=TSTVL1,DISP=OLD
//TAPE1       DD  SUBSYS=SOV,DSN=DUMP.FDRDSF.TSTVL1(0)
//SYSPRIN1    DD  SYSOUT=*
//SYSIN       DD  *
      RESTORE   TYPE=DSF,DATA=ALL,DSNENQ=TEST,MAXERR=1
      SELECT    ALLDSN
/*
//RESJOB2     JOB jobcard info
//STEP001     EXEC  PGM=FDRDSF,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//DISK1       DD  UNIT=SYSDA,VOL=SER=TSTVL2,DISP=OLD
//TAPE1       DD  SUBSYS=SOV,DSN=DUMP.FDRDSF.TSTVL2(0)
//SYSPRIN1    DD  SYSOUT=*
//SYSIN       DD  *
```

```
     RESTORE   TYPE=DSF,DATA=ALL,DSNENQ=TEST,MAXERR=1
     SELECT    ALLDSN
/*
//RESJOB3     JOB jobcard info
//STEP001     EXEC  PGM=FDRDSF,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//DISK1       DD  UNIT=SYSDA,VOL=SER=TSTVL3,DISP=OLD
//TAPE1       DD  SUBSYS=SOV,DSN=DUMP.FDRDSF.TSTVL3(0)
//SYSPRIN1    DD  SYSOUT=*
//SYSIN       DD  *
     RESTORE   TYPE=DSF,DATA=ALL,DSNENQ=TEST,MAXERR=1
     SELECT    ALLDSN
/*
```

Basic ExHPDM FDRTCOPY in Parallel

The following example demonstrates how to perform FDRTCOPY with ExHPDM. Refer to “Basic ExHPDM Backup” on page 79 for the original dump. Multiple jobs must be run to make FDRTCOPY processing run in parallel. If the jobs are not run in parallel, then streaming of the data by ExHPDM does not occur resulting in a longer overall elapse time.

The use of the SELECT(OFFSITE) CLASS(OFFSITE) rule is being used to direct the copied data to the OFFSITE stream. WAIT(NO) is specified on the TAPEIN DDs to avoid possible deadlocks when the input stream device is unavailable. WAIT(NO) is specified on the OFFSITE stream for the same reason. CATCHUP(ALWAYS) is specified on the input data sets to force all clients to be read on the same pass of the stream file. If this was not specified, then it is possible that one or two of the clients would have to wait if their first block was already passed by the time they connect to the stream task.

```
//DMPCPY1  JOB jobcard info
//STEP001     EXEC PGM=FDRTCOPY,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//TAPEIN      DD  SUBSYS=(SOV,'WAIT(NO) CATCHUP(ALWAYS)'),
//DSN=DUMP.RD3A.FULL.HPDM01(0)
//TAPEOUT     DD  SUBSYS=SOV,
//               DSN=DUMP.RD3A.OFFSITE.HPDM01(+1),
//               DISP=(,CATLG,DELETE)
//SYSIN       DD  *
  COPY
//DMPCPY2  JOB jobcard info
//STEP001     EXEC PGM=FDRTCOPY,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//TAPEIN      DD  SUBSYS=(SOV,'WAIT(NO) CATCHUP(ALWAYS)'),
//DSN=DUMP.RD3A.FULL.HPDM02(0)
//TAPEOUT     DD  SUBSYS=SOV,
//               DSN=DUMP.RD3A.OFFSITE.HPDM02(+1),
//               DISP=(,CATLG,DELETE)
//SYSIN       DD  *
  COPY
//DMPCPY3  JOB jobcard info
//STEP001     EXEC PGM=FDRTCOPY,REGION=4M
//SYSPRINT    DD  SYSOUT=*
//TAPEIN      DD  SUBSYS=(SOV,'WAIT(NO) CATCHUP(ALWAYS)'),
//DSN=DUMP.RD3A.FULL.TSO189(0)
//TAPEOUT     DD  SUBSYS=SOV,
//               DSN=DUMP.RD3A.OFFSITE.TSO189(+1),
```

```
//              DISP=(,CATLG,DELETE)
//SYSIN         DD  *
  COPY
```

ExHPDM with FDRABR Backup

Using ExHPDM and FDRABR is identical to using FDRABR without ExHPDM. Backing up data with ExHPDM is seamless and invisible after the JCL is modified to form the connection between the JCL DDNAME statement and the ExHPDM server. To generate as many parallel dumps as possible using FDRABR, nine TAPE*x* DDs should be specified. This indicates to FDRABR to have nine parallel dump tasks active.

The job in the following example performs nine concurrent full volume (TYPE=FDR) physical backups of volumes that start with the volser HPDM0 (VOLG=HPDM0). The DD statements TAPE1 to TAPE9 indicate to FDRABR that nine dumps should be active in parallel. The client data set names are generated by FDRABR when the TAPE*x* DDs are opened and conform to the FDRABR naming standard. The only specification required on the TAPE*x* DDs is the SUBSYS=SOV specification indication that the DDs should be processed by the ExHPDM server whose subsystem name is SOV. These volumes are backed up to the stream called REDST3A on the tape device known by the esoteric name HPDMRW. This is due to the processing by the SELECT statement REDRU3A. The server writes these data sets to a stream file on tape whose name is prefixed by 'HPDM.HPDMD3A.STREAM.'

```
//DUMPJOB    JOB jobcard info
//STEP001     EXEC  PGM=FDRABR,REGION=0M
//SYSPRINT    DD  SYSOUT=*
//TAPE1       DD  SUBSYS=SOV
//TAPE2       DD  SUBSYS=SOV
//TAPE3       DD  SUBSYS=SOV
//TAPE4       DD  SUBSYS=SOV
//TAPE5       DD  SUBSYS=SOV
//TAPE6       DD  SUBSYS=SOV
//TAPE7       DD  SUBSYS=SOV
//TAPE8       DD  SUBSYS=SOV
//TAPE9       DD  SUBSYS=SOV
//SYSPRIN1    DD  SYSOUT=*
//SYSPRIN2    DD  SYSOUT=*
//SYSPRIN3    DD  SYSOUT=*
//SYSPRIN4    DD  SYSOUT=*
//SYSPRIN5    DD  SYSOUT=*
//SYSPRIN6    DD  SYSOUT=*
//SYSPRIN7    DD  SYSOUT=*
//SYSPRIN8    DD  SYSOUT=*
//SYSPRIN9    DD  SYSOUT=*
//SYSIN       DD  *
 DUMP TYPE=FDR,MAXERR=1
 MOUNT VOLG=HPDM0
/*
```

To perform incremental backups of data sets contained on the selected volumes the TYPE=ABR operand is used. The following job selects the same volumes used in the previous JCL example to perform an incremental backup. The data set name created for each volume backup is generated by FDRABR during open processing. An incremental

backup results in a new cycle being created. The same SELECT, CLASS, STREAM and DEVICE definitions apply to this job.

```
//DUMPJOB   JOB jobcard info
//STEP001   EXEC  PGM=FDRABR,REGION=0M
//SYSPRINT  DD  SYSOUT=*
//TAPE1     DD  SUBSYS=SOV
//TAPE2     DD  SUBSYS=SOV
//TAPE3     DD  SUBSYS=SOV
//TAPE4     DD  SUBSYS=SOV
//TAPE5     DD  SUBSYS=SOV
//TAPE6     DD  SUBSYS=SOV
//TAPE7     DD  SUBSYS=SOV
//TAPE8     DD  SUBSYS=SOV
//TAPE9     DD  SUBSYS=SOV
//SYSPRIN1  DD  SYSOUT=*
//SYSPRIN2  DD  SYSOUT=*
//SYSPRIN3  DD  SYSOUT=*
//SYSPRIN4  DD  SYSOUT=*
//SYSPRIN5  DD  SYSOUT=*
//SYSPRIN6  DD  SYSOUT=*
//SYSPRIN7  DD  SYSOUT=*
//SYSPRIN8  DD  SYSOUT=*
//SYSPRIN9  DD  SYSOUT=*
//SYSIN     DD  *
 DUMP TYPE=ABR,MAXERR=1
 MOUNT VOLG=HPDM0
/*
```

ExHPDM with FDRABR Restore

When using FDRABR to restore data using ExHPDM, no actual reference to the ExHPDM subsystem is required. When using the FDRABR DYNTAPE operand, allocations are performed by FDRABR to ExHPDM where the backups were originally performed with ExHPDM. FDRABR, by default, uses the ExHPDM subsystem name SOV. If a subsystem name other than SOV is required, then contact Innovation Data Processing. Where more than one volume restore is to be performed in parallel, multiple jobs must be run.

The following example job performs four parallel restores for volumes HPDM01-HPDM04. These restores are using the incremental backup data created from the previous example “ExHPDM with FDRABR Backup” on page 83. When performing restore processing, it is preferable to submit jobs for as many restores as is possible. This allows ExHPDM to select the stream files required and stream the data effeciently. If jobs are not submitted correctly, it is possible that multiple reads of the same stream file are required.

```
//RESTJOB1  JOB jobcard info
//STEP001   EXEC  PGM=FDRABR,REGION=4M
//SYSPRINT  DD  SYSOUT=*
//DISK1     DD   VOL=SER=HPDM01,UNIT=SYSALLDA,DISP=OLD
//SYSIN     DD   *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=HPDM01
/*
```

```
//RESTJOB2   JOB jobcard info
//STEP001    EXEC  PGM=FDRABR,REGION=4M
//SYSPRINT   DD  SYSOUT=*
//DISK2      DD   VOL=SER=HPDM02,UNIT=SYSALLDA,DISP=OLD
//SYSIN      DD   *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=HPDM02
/*
//RESTJOB3   JOB jobcard info
//STEP001    EXEC  PGM=FDRABR,REGION=4M
//SYSPRINT   DD  SYSOUT=*
//DISK3      DD   VOL=SER=HPDM03,UNIT=SYSALLDA,DISP=OLD
//SYSIN      DD   *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=HPDM03
/*
//RESTJOB4   JOB jobcard info
//STEP001    EXEC  PGM=FDRABR,REGION=4M
//SYSPRINT   DD  SYSOUT=*
//DISK4      DD   VOL=SER=HPDM04,UNIT=SYSALLDA,DISP=OLD
//SYSIN      DD   *
 RESTORE TYPE=FDR,MAXERR=1,CONFMESS=NO,DYNTAPE
 SELECT VOL=HPDM04
/*
```

Listing the ExHPDM Database

The following example demonstrates how to list the contents of the database using the SOVADMN utility. There are a number of variations for the ADMIN DATABASE LIST command. Note that mixed case can be used for SOVADMN commands and each command must start on a new line.

```
//SOVADMN  JOB jobcard info
//*  LIST THE CONTENTS OF THE EXHPDM DATABASE
//ADMLST     EXEC PGM=SOVADMN
//STEPLIB    DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT   DD  SYSOUT=*       utility messages here.
//SYSTERM    DD  SYSOUT=*       used for diagnostic messages.
//SNAMSOV    DD  DUMMY       directs requests to ExHPDM server.
//SYSIN      DD  *
 admin database list
 admin database list client(dsn(CLIENT.RD3A.FULL.HPDM01))
 admin database list client(OWNER(JRA) DSN(**))
 admin db list s(REDST3A file(**))
 admin Db list s(REDST3A file(**)
                 creationdate(08oct1997 21oct1997))
 admin db list stream(** file(CLIENT.RD3A.**)
                 creationdate(08jun1998 09dec1998))
 admin db list s(REDST3A
                 file(HPDM.HPDMD3A.STREAM.WUTB2B53))
/*
```

Deleting from the ExHPDM Database

The following example demonstrates how to use the SOVADMN utility to delete specific contents of the database. There are numerous options to the ADMIN DATABASE DELETE command. Note that mixed case can be used for SOVADMN commands.

```
//SOVADMN  JOB jobcard info
//* Delete the contents of the ExHPDM database
//ADMNDEL   EXEC PGM=SOVADMN
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT  DD  SYSOUT=*        utility messages here.
//SYSTERM   DD  SYSOUT=*   used for diagnostic messages.
//SNAMSOV   DD  DUMMY directs requests to ExHPDM server.
//SYSIN     DD  *
 admin db delete c dsn(CLIENT.RD3A.FULL.HPDM01) gen(all)
 admin database delete s file(HPDM.HPDMD3A.STREAM.WUTB2B53)
 admin database delete stream file(**)
                crdt(01jan1997 01apr1997)
admin database delete s file(HPDM.HPDMD3A.STREAM.TWOB0RN0)
                creationdate(03may1997 04may1998)
/*
```

Deleting Expired Client Data Sets from the ExHPDM Database

The following example demonstrates how to use the SOVADMN utility to delete expired client data sets from the database. The management definition UNCAT for the example definitions indicates that client data sets expire when they are no longer cataloged. This management definition is used by CLASSes REDCL3A, RECL3E and TLCLASS. The management definition KEEP10 indicates that client data sets expire after 10 days. This management definition is used by CLASS FDR1.

```
//SOVADMN  JOB jobcard info
//* Delete expired clients data sets from the ExHPDM database
//ADMNDEL   EXEC PGM=SOVADMN
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINT  DD  SYSOUT=*        utility messages here.
//SYSTERM   DD  SYSOUT=*   used for diagnostic messages.
//SNAMSOV   DD  DUMMY directs requests to ExHPDM server.
//SYSIN     DD  *
 admin scanexpire
/*
```

Using SCANSTREAMFILE from SOVADMN

The following example demonstrates how to use the SCANSTReamfile command to build a volatile database for use to restore the contents of some ExHPDM stream files. The ExHPDM server is started in DR mode. Note that the SCANSTReamfile command must only be used when the ExHPDM database cannot be recovered.

Note: In order to provide enough time for the job to finish and not be terminated by the system, the SCANSTReamfile command requires a **TIME=1440** or **TIME=nolimit** parameter.

The first ADMIN SCANSTReamfile command builds entries in the volatile database for all contents of the valid stream file contained in the volumes 542628, 542631, 542632, and 542570.

The second ADMIN SCANSTReamfile command builds the contents of the valid stream file contained in volume 542503 and stops scanning with it has successfully built an entry for the client file named **CLIENT.HPDMA1.COMP30PC.DUMPDEC1.**

The third ADMIN command builds a list of the volatile database. This list should be used as reference for any subsequent restore processing.

```
//SOVADMNJOB jobcard info
//* Build a volatile database with SCANSTR
//ADMNSSF EXEC  PGM=SOVADMN
//STEPLIBDD  DISP=SHR,DSN=EXHPDM.STKLOAD
//SYSPRINTDD  SYSOUT=*          utility message here.
//SYSTERMDD  SYSOUT=*   used for diagnostic messages.
//SNAMSOVDD  DUMMY directs requests to ExHPDM server.
//SYSINDD  *
 ADMIN SCANSTR VOL(542628 542631 542632 542570)
               UNIT(CART)
 ADMIN SCANSTR VOL(542503) UNIT(3490)
               COMPLETE(FVALID(DSN(CLIENT.HPDMA1.COMP30PC.DUMPDEC1)))
 ADMIN DATABASE LIST
/*
```

Chapter 6. ExHPDM Startup Parameter File

Overview

This chapter describes the keywords and parameters used in the ExHPDM startup parameter file. The contents of the startup parameter file are used when ExHPDM is first started and whenever the SET PRM operator command is issued. The overall format of the parameter file is described by the syntax diagram “ExHPDM Startup Parameter File Format” on page 91.

The ExHPDM samplib library (STKSAMP) contains a number of examples of the startup parameter file. A simple example that meets most needs is contained in the member SOVPRM00. This example can be used, but must be customized for each MVS system environment. The startup parameter file supports many options. In most cases, the options have reasonable defaults. A useful startup parameter file may be as small as ten lines.

ExHPDM operator commands are described in “Chapter 8. ExHPDM Operator Commands” on page 183.

At the end of this chapter, is a section called, “Startup Parameter File Example.” This section contains a series of examples that tie together most of the keywords in the ExHPDM startup parameter file.

See “ Scope” on page 3 for the syntax conventions used in this section.

Note: Parenthesis are mandatory in the startup parameter file and must be entered as indicated in the syntax diagrams.

Defining the Startup Parameter File

The ExHPDM server uses the parameter file on startup to define the way that it should be run. This parameter file is stored and used on the MVS system(s) where ExHPDM is started. The startup parameter file may be stored in the MVS parameter library SYS1.PARMLIB or a user defined library as specified in the ExHPDM server startup procedure.

The ExHPDM server parameters are contained, by default, in the partitioned data set member SOVPRM*xx* in the MVS parameter library SYS1.PARMLIB. The two character suffix (*xx*) on the data set member name, SOVPRM*xx*, enables a number of parameter members to be defined. The suffix is specified when ExHPDM is started through the PRM(*xx*) keyword in the MVS START command or the IEFSSN*xx* parmlib member. If the

PRM keyword is not specified, PRM defaults to PRM(00). The MEMBER keyword may also be used to change the member name prefix from the default of SOVPRM.

If the STKPARMS DD name is defined in the ExHPDM started task PROC, ExHPDM reads the parameters from it instead of from the SYS1.PARMLIB member. If the STKPARMS DD name points to a partitioned data set, the MEMBER and PRM keywords are used to determine the member in the same manner as for the SYS1.PARMLIB member (the default is SOVPRM00). The STKPARMS DD name may also point to a sequential data set, in which case the parameters are read from the data set directly and cannot be dynamically refreshed. If the STKPARMS file cannot be opened for any reason, the SYS1.PARMLIB member is used.

Most parameter file changes made while ExHPDM is active can be refreshed through the use of the SET PRM operator command. See “PRM” on page 177 for a description.

Note: DATABASE and REQUEST changes do not take effect until the system is restarted after using the SET PRM operator command. See “DATABASE Keyword” on page 102 and “REQUEST Keyword” on page 130

The SOVPRM*xx* parameter file is specified in the following format. The substitution block **DB Optional Parameters** is expanded by “DATABASE Keyword” on page 102.

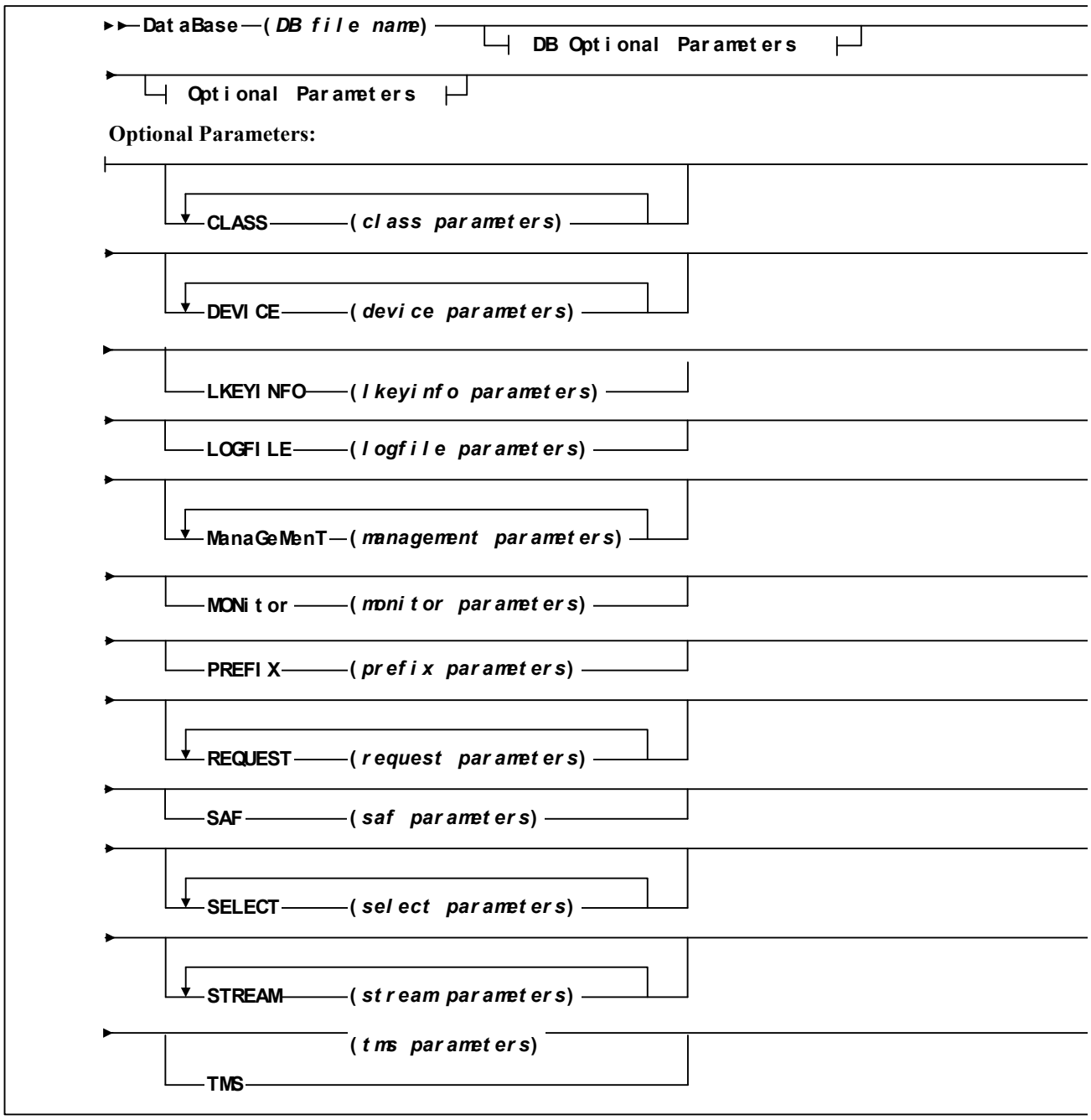


Figure 8. ExHPDM Startup Parameter File Format

The SOVPRM*xx* parameter file has the following characteristics:

- Parameters and commands passed to ExHPDM are processed by the StorageTek Common Parser (FMID SSKQ250).
- The ExHPDM parameter file can be a sequential data set or a PDS member. It can have a record format of F (fixed) or V (variable) or FB (fixed block) or VB (variable

block) with a logical record length (LRECL) up to 1024. The block size can be any block size acceptable to the device.

- The ExHPDM parameter file may contain mixed case. This includes system symbol specifications.

Note:

1. ISPF PACK data set format is not supported for the ExHPDM parameter file. When editing the parameter file ensure that PACK OFF is specified in the ISPF edit profile.
2. Parenthesis are mandatory in the startup parameter file and must be entered as indicated in the syntax diagrams.
3. The parser processes input in a variety of forms and is not limited to the conventional 80-byte card input. All input columns are scanned.Text editor line numbers, which are traditionally used in columns 73 through 80, should not be used because these are interpreted by the parser as errors.

The following sections describe the user input requirements and how they are specified for use with ExHPDM.

Sharing Parameter File Definitions Using System Symbols

This section describes how to set up the SOVPRM*xx* definition so two or more systems, running ExHPDM, can share a common SOVPRM*xx* definition.

Various parameters in the ExHPDM SOVPRM*xx* definition must be unique to each executing occurrence of an ExHPDM server. These parameters are:

- JouRNaL(*journal file name*)
- LOGFILE(DSN(*log file name*)).

However, many of the other options contained in the ExHPDM SOVPRM*xx* definition are applicable to all ExHPDM servers.

To allow multiple use of the ExHPDM SOVPRM*xx* definition, system symbols introduced in IBM MVS Version 5.1 can be used on any parameters in the parameter definition. By using system symbols, it is possible to generate unique parameter values (that is, log file and journal data set names) and allow the ExHPDM parameters to be shared. For example, the following shows the usage of the system name (&SYSNAME) in the JouRNal and LOGFILE data set specifications:

```
JouRNal(EXHPDM.JRNL.&SYSNAME)
LOGFILE(DSN(EXHPDM.LOG.&SYSNAME APPEND))
```

Symbols can be used in any location in the SOVPRM*xx* data set.

MVS allows the use of two different types of symbols:

- Static symbols
- Dynamic symbols.

Static symbols are those that do not change from IPL to IPL. The MVS static symbols can be displayed through the use of the MVS MCS console DISPLAY SYMBOLS command. The present set of standard MVS static symbols are:

- &SYSNAME (system name)
- &SYSCLONE (last 2 characters of system name)
- &SYSPLEX (sysplex name)
- &SYSR1 (IPL volume serial name).

Further static symbols, defined by the user, can be specified through the IEASYM*xx* parmlib member, SYMDEF directive.

Dynamic symbols are those that change depending on the current status of some event. For example, &JOBNAME, &DATE, &DAY, etc. All these symbols and the use of IEASYM*xx* are described in detail in the publication, *IBM MVS/ESA SP V5 Initialization and Tuning Reference.*

In addition to the standard MVS static and dynamic symbols, the following ExHPDM defined symbols can be used:

- &SOVSSN (ExHPDM subsystem name)
- &SOVPID (ExHPDM product identifier 'SOV')
- &SOVFMID (ExHPDM FMID).

Symbolic substitution is performed whenever ExHPDM is started and when the ExHPDM SET PRM operator command is issued.

Note: System symbols can also be used in the stream file name, such that the symbols are resolved in the name at the time a new stream file is created. See "STREAM Keyword" on page 142 for details about specifying symbols in the stream file name.

When text substitution is performed for a particular parameter line, message SOV03027I is issued. This message is similar to the MVS message IEE295I issued during symbolic substitution of operator commands. Message SOV03027I shows the original parameter text and the modified parameter text:

SOV03027I Parameter text changed by symbolic substitution:
Original : ***original parameter text.***
Modified : ***modified parameter text.***

Customizing the ExHPDM Parameter File

The following should be considered when customizing the ExHPDM parameter file. With the exception of the first bullet item, which is mandatory, the remaining items are optional.

Note: SELECT, CLASS, STREAM, and DEVICE keywords must be given very detailed attention because they are the heart of ExHPDM's ability to perform and meet your needs.

- ExHPDM's database name in the DATABASE keyword. This must be defined so all ExHPDM servers can find it. Because the database is a VSAM file, it must be found in the MVS catalog.
- ExHPDM journal file name in the JouRNaL keyword. If journaling is required, the journal name must be defined for each ExHPDM server and must be unique to that server. This name can contain any valid system symbols to create a unique data set name.
- ExHPDM space monitoring of the ExHPDM database and journal files using the MONITOR keyword.
- ExHPDM log file destination in the LOGFILE keyword. The log file is where the ExHPDM server logs its messages. If LOGFILE is not specified, ExHPDM defaults to log all messages to consoles with route code 11.
- TMS specification in the TMS keyword. This allows ExHPDM to interface to the installed TMS to return tape volumes.
- ExHPDM command and message prefixes in the PREFIX keyword.The default values are normally adequate; however, in situations where it is preferable to identify (e.g., messages from particular ExHPDM servers), a user-defined value may be specified.
- Security specification in the SAF keyword. ExHPDM interfaces with the system security product (e.g., RACF) using SAF. ExHPDM validates a client's ability to use particular ExHPDM resources.
- Select rules in the SELECT keyword. Select rules are a set of rules that can be used by the user to allow connection to ExHPDM.
- Class definitions in the CLASS keyword. A stream class is specified by the client, in its JCL, or by the server (using select rules) to direct a connection to a particular ExHPDM stream. There are examples of stream class definitions in the samplib members on the installation base tape. These must be customized to fit the MVS environment in which ExHPDM is operating.
- Management definitions in the ManaGeMenT keyword. Stream management is used to define how a client data set is managed once it is created. The management for the client is specified by the client, in its JCL, by select rules or by the stream class.
- Stream definitions in the STREAM keyword. Streams can be defined to meet the needs of the ExHPDM environment. The stream file data set name prefix must be specified and is used to form fully qualified stream file data set names.
- Device definitions in the DEVICE keyword. The device definitions must be customized to match the actual tape devices in the MVS system.
- License Key definitions in the LKEYINFO keyword. These definitions must be entered or ExHPDM will not run.

Note: You must use esoteric device names or generic device names in ExHPDM device definitions at all times.

ACTIVATE Keyword

The ACTIVATE keyword allows the specification of two options.

The first option allows the cataloging of a client that has not completed successfully and provides a mechanism to automatically delete clients at the end of a step. This value will be used if no disposition is coded on the client's DD statement.

The second option allows you to determine from which ExHPDM server a message is being issued when there are multiple servers active on the one system.

Syntax

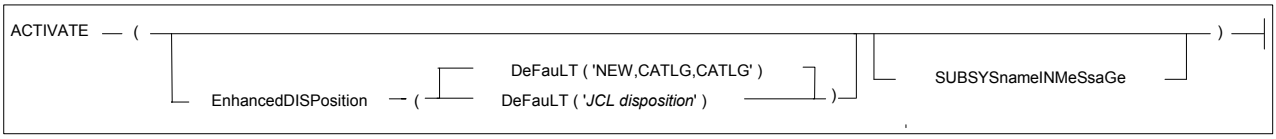


Figure 9. ACTIVATE Parameters

Keyword Name

ACTIVATE
initiates the **ACTIVATE** keyword. It allows the specification of two options.

Parameters

EnhancedDISPosition
indicates that enhanced disposition processing will be active. If this is not specified then the current DISP scheme will be honored.

No additional keywords for the client are required to allow the specification of the DISP parameters. These are part of the MVS JCL, and DYNALLOC specification on the DD statement.

Note: Catalog entries in the MVS ICF catalog structure are only deleted, where indicated, when the catalog entry indicates that it belongs to ExHPDM. ExHPDM catalog entries are identified by the first volume being set to **SOV*. Where a data set is cataloged which does not have **SOV* as an identifying volume name then ExHPDM will not uncatalog the data set. However, other processing indicated by the uncatalog processing will still be performed.

The format of the DISP statement is as follows:

```
DISP=(status,normal,abnormal)
```

A default DISPosition can be specified. This keyword is a mandatory field that must be specified.

status

is the disposition status. This value is not used by ExHPDM directly. However, it may influence the defaults for the normal and abnormal disposition values. These defaults are set by the MVS JCL processing and are documented in the JCL reference manual.

normal

ExHPDM will catalog the client data set when CATLG or KEEP are specified or defaulted.

ExHPDM will not catalog the client data set when UNCATLG, PASS or DELETE are specified or defaulted.

For read processing only, ExHPDM will uncatalog the client data set when UNCATLG and ACTIVATE(ENHANCEDDISPOSITION) are specified.

For read processing only, ExHPDM will set the client data set as not valid (i.e deleteable) in the ExHPDM data base when DELETE and ACTIVATE(ENHANCEDDISPOSITION) are specified.

abnormal

The abnormal disposition is only honoured when ACTIVATE(ENHANCEDDISPOSITION) is specified and the client completes prematurely (e.g abend, ExHPDM cancel etc). Otherwise, client data sets are only set as valid if they complete successfully. This is equivalent to DISP=(,,DELETE).

For write processing only, ExHPDM will catalog the client data set when CATLG or KEEP are specified or defaulted. This allows the client data set to be accessed as valid data even when the client job has terminated abnormally.

ExHPDM will not catalog the client data set when UNCATLG or DELETE are specified or defaulted.

For read processing only, ExHPDM will uncatalog a client data set when UNCATLG is specified or defaulted.

For read and write processing, ExHpdM will not catalog the client data set and it will be set as not valid (i.e deleteable) in the ExHPDM data base when DELETE is specified.

JCL disposition

Note: Care needs to be taken regarding the default disposition entered. A default disposition of NEW,DELETE,DELETE will result in the deletion of customer data, both on normal and abnormal completion.

NEW,CATLG,CATLG

allows a default value to be set for the disposition. If the enhanced disposition is active but no default value is given, then it is assumed to be NEW,CATLG,CATLG.

SUBSYSnameINMeSsaGe

provides the ability to determine which ExHPDM subsystem is processing the client request when multiple ExHPDM's are active on the same system.

ACTIVATE Keyword Examples

The output of message SOV06900I allows the ACTIVATE field value to be displayed.

Example 1

The following example shows the possible output displayed for the ACTIVATE keyword where the associated parameter E ENHANCEDDISPOSITION(DEFAULT('NEW,CATLG,CATLG')) value has been specified, but no default has been specified.

```
9.59.03 IEESYSAS  SOV06900I Parameter file is DSN:SYS1.PARMLIB(SOVPRM00)
Parameter   (O)riginal Parameter file value
        (E)ffective value
        (C)hange value ; in progress
------------ - ------------------------------------------------------
..
..
ACTIVATE     O ENHANCEDDISPOSITION
         E ENHANCEDDISPOSITION
..
..
```

Example 2

The following shows the possible output displayed for the ACTIVATE keyword where no values have been specified.

```
T19.59.03 IEESYSAS  SOV06900I Parameter file is DSN:SYS1.PARMLIB(SOVPRM00)
Parameter   (O)riginal Parameter file value
        (E)ffective value
        (C)hange value ; in progress
------------ - ------------------------------------------------------
..
..
ACTIVATE     O Not specified
..
..
```

Additional messages to the client to indicate the client disposition values and whether or not the client data set was UNCATLGd or DELETEd. These messages are accessible in the client's JESSYSMSG data set as well as the ExHPDM logfile. ExHPDM indicates the catalog status at close time via messages SOV06217I and SOV06218I.

Example 3

Message SOV06031I shows the disposition values that are being used for the client data set. These are shown in bold:

SOV06031I Connection ID *%s* for client *%s*:
data set *%s* using stream *%s*,
RECFM=*%s* BLKSIZE=*%s* LRECL=*%s* BUFNO=*%s*
UNIQUIFER= %08X%08x, NORMAL=*%s*, ABNORMAL=*%s*

Example 4

A message indicates when the data set is deleted. This means that the client data set is not valid and cannot be processed on subsequent read processing:

SOV06xxxI Client *%s*, DSN *%s* DELETED.

A message indicates when the data set is uncataloged. The client data set is still accessible, as it is still valid. Only the MVS catalog entry is deleted:

SOV06xxxI Client *%s*, DSN *%s* UNCATALOGED.

Example 5

A message indicates that subsystem, SOV1, issued the SOV05005 message:

SOV050051 (SOV1) ExHPDM Log Message Manager has shut down.

CLASS Keyword

The CLASS keyword is only used for client write requests. It is an optional keyword that defines the performance and management attributes of the client data and it allows the grouping of this client data to a particular stream definition. That is, it is a specification for a class of data. The CLASS is selected when a client write connection is requested. The *stream class name* is selected by either the SELECT CLASS or the DD SUBSYS CLASS keyword as described by "SELECT Keyword" on page 134 and "Both BACKUP and RESTORE steps may require some time to complete. A 300-cylinder database may take between one and 20 minutes to restore, depending on system speed and other workloads. Even starting ExHPDM with a new (empty) database may take a few minutes, since the database needs to be formatted before use." on page 272. The usage of the class definition is described in "Identify the Performance Attributes for the Client Connection" on page 37.

Syntax

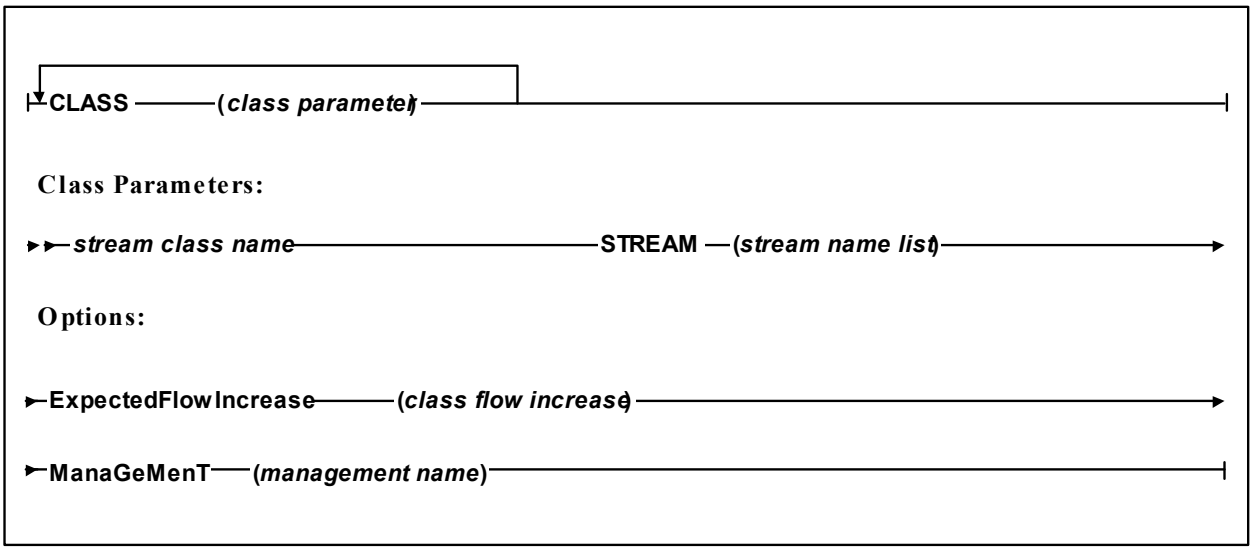


Figure 10. CLASS Keyword

Keyword Name

CLASS
initiates the CLASS keyword and specifies a unique stream class name. The *stream class name* can be up to 8 characters and must begin with an alphabetic character.

Parameters

STREAM
is a required parameter that specifies a list of stream names that indicate which stream definition this class can use. The list of streams specified by *stream name* is used in conjunction with the limits specified in the FLOWLIMIT**(***device flow limit***)** parameter of the DEVICE keyword and/or the CONNECTIONS (*max stream*

connections) parameter of the STREAM keyword to select a currently active stream definition or to start a new occurrence of a stream definition**.** Up to 255 *stream name* entries can be specified.

ExpectedFlowIncrease
specifies the expected performance level that a connection to this stream class adds to the class flow for a particular stream definition. *class flow increase* can be specified in bytes per second (3500000), KB per second (3500K), MB per second (3.5M), or GB per second (.0035G). If a value is specified that is less than 1 KB per second, then the value is ignored.

ManaGeMenT
stream management name specifies the management name to assign to clients using this class. This value is not used if a ManaGeMent is specified in the associated SELECT or DD SUBSYS parameters or an EXPDT or RETPD was specified on the client connection JCL.

CLASS Keyword Examples

The following are examples of the CLASS keyword.

Example 1

The following example defines a stream class for operation's MVS operating system backup work. This stream class uses stream TLSTRM.

```
CLASS(OPERPROD STREAM(TLSTRM) )
```

Example 2

The following example defines a stream class for all DB2 backup work. Each connection is assumed to be capable of delivering 3 MB per second from the disk arrays.

```
CLASS(DB2PROD STREAM(TLDSSTRM) EXPECTEDFLOWINCREASE(3M) )
```

Example 3

The following example defines a stream class for LAN backup work. Load balancing is not used as the default value for ExpectedFlowIncrease is 0. The STREAM CONNECTION parameter is used in this instance to provide load balancing.

```
CLASS(LANBACK STREAM(TLSTRM) )
```

Example 4

The following example defines a stream class for backup work. Load balancing as indicated by the EFI(0) value is not used by this class of data. The STREAM CONNECTION parameter is used in this instance to provide load balancing.

```
CLASS(DBAPROD EFI(0) STREAM(DBASTRM) )
```

Example 5

The following example defines a stream class to be used for all IMS production backups. These applications are backed up using Redwood type ’A’ cartridges (10 MB). Load balancing is not used.

```
CLASS(IMSPROD STREAM(REDST3A) )
```

Example 6

The following example defines a stream class to be used for clients requiring medium amounts of high-capacity storage. Clients are assumed to be capable of delivering 1500 KB per second from the disk arrays.

```
CLASS(REDCL3B STREAM(REDST3B) EFI(1500K) )
```

Example 7

The following example defines a stream class used for seismic data archiving that requires high-performance and high-capacity Redwood type ’c’ cartridges (50 GB). It is expected that the clients are capable of delivering 7 MB per second from a server’s disk array. A list of stream names is specified.

```
CLASS(SEISMIC EFI(7M) STREAM(REDST3C REDST3D REDST3E))
```

DATABASE Keyword

The ExHPDM database is specified to the ExHPDM server through the use of the DATABASE keyword. This is the only mandatory parameter in the startup parameter file.

Caution: The data set name specified for the **JouRnaL (***journal file name***)** is not verified for existence until the database is successfully started. Additionally, the values specified for the **BACKupJouRNal**, **BACKupDSN**, and **UNIT** parameters are verified only for data set name and unit syntax. The DFP correctness is verified only when the parameters are used.

Note: Changes to database or journal file names do not take effect until ExHPDM is restarted.

If an uncataloged or incorrect VSAM entry is used, the database starts in quiesced mode.

The database may be allocated using the sample job in the SOVDBDEF member of the STKSAMP installation partitioned data set. Refer to "Considerations for Allocating the ExHPDM Database" on page 268 when creating the database

Syntax.

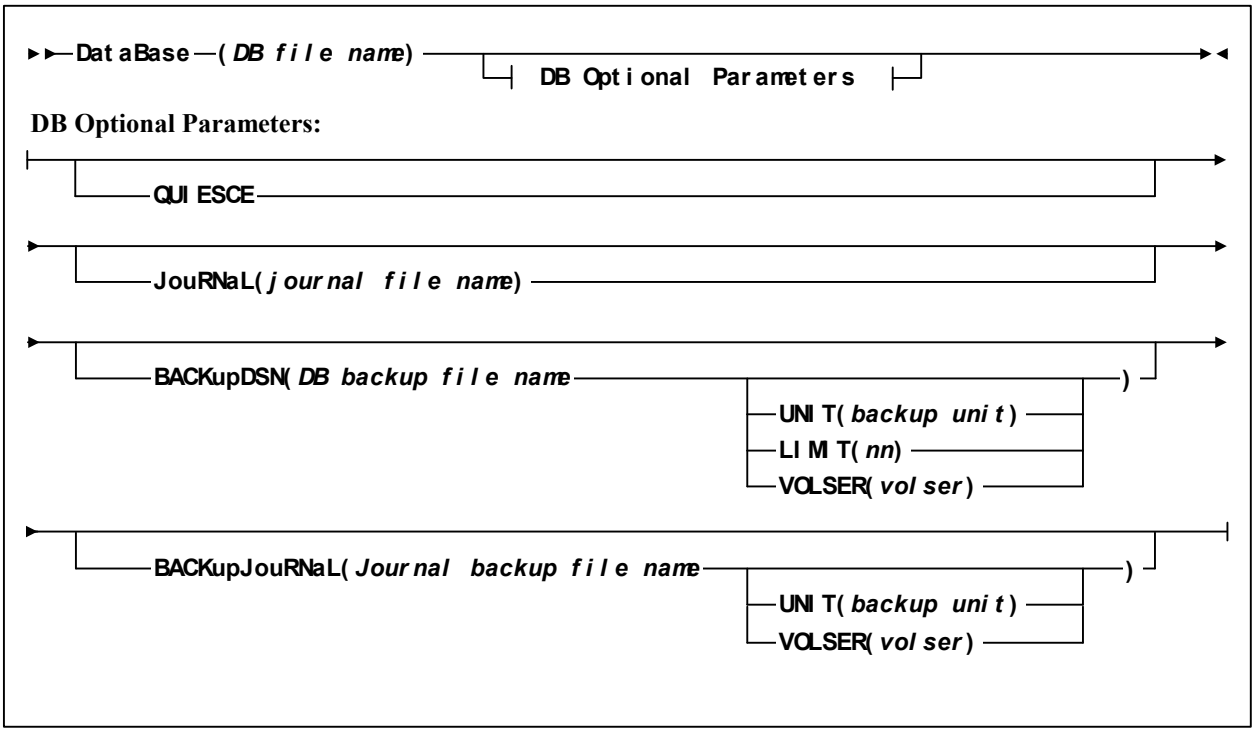


Figure 11. Database Keyword

Keyword Name

DataBase

initiates the DATABASE keyword and specifies the VSAM KSDS database MVS data set name.

Parameters

QUIESCE

specifies that when ExHPDM is started, the database is not allocated. No new ExHPDM connections can be created without the database. Database recovery command operations can only take place while the database is quiesced. The database can also be quiesced by using the SET **DATABASE QUIESCE** operator command.

JouRNaL

journal_file_name specifies the name of the sequential dataset that ExHPDM uses as its journal dataset.

BACKupDSN

DB backup file name specifies the data set name prefix that is used to backup the database described by **DataBase(***DB file name***).** The *DB backup file name* prefix can be up to 35 characters long. The backup of the database is performed using the SOVADMN utility DATABASE BACKUP command. The *DB backup file name* is suffixed during DATABASE BACKUP processing with V*xxxxxxx*, where *xxxxxxx* is the current backup version of the database. This value is the same as the value used to backup the journal. The file is allocated to the unit described by the **UNIT** parameter.

This parameter must be specified to perform database backups.

Database and Journal Backup dataset count is defined by the user in the ExHPDM parameter file by use of the LIMIT subparameter of the BACKupDSN parameter.

Warning: If journaling is used, ensure that the journal is equal to one-third the size of the database so that the journal is large enough to include all required updates. If the journal fills up during the compress processing, cancel the current backup job and ExHPDM server and restart and resubmit the backup job. It is recommended that the customer perform another backup after the initial backup is completed to clear the journal contents and create a backup of the database in compressed form.

UNIT

backup unit specifies the destination of the backup files on DASD. SMS might influence the allocation of the journal backup files. *backup unit* should only contain DASD devices as random access to *journal backup file name* is required during backup processing.

LIMIT

backup limit is used to determine how many database and journal backups to keep. If the backup limit is zero (0) or is not specified, the current functionality prevails, i.e. ExHPDM does not manage the number of backup datasets and does not delete old backup datasets.

If non-zero (1 or more), ExHPDM uses the backup limit as follows: 1) It subtracts the limit value from the newly created version number to calculate a starting version number for the delete process. 2) It then deletes the database and journal backups with that starting version number and all older versions, i.e.

those backups whose version number is less than the starting value. 3) Due to the way the restore process works, ExHPDM also deletes the journal backup that is one or more than the starting version number. **Note:** The deletion of this additional journal backup means that if the limit is one (1), then no journal backups are kept.

New messages have been written to the ExHPDM SOVADMN output data set (SYSPRINT) indicating the action taken during LIMIT processing.

The output for the DISPLAY OPTIONS operator command indicates the database limit values.

For example, the following output from a DISPLAY OPTIONS command shows a LIMIT of 5 was specified:

```
19.59.03 IEESYSAS  SOVO69OO1  Parameter file is DSN:SYS1.PARMLIB(SOVPRMOO
Parameter          (O)riginal Parameter file value
                   (E)ffective value
                   (C)hange value : in progress
---------          ------------------------------------------------------------

..
BACKDSN            O DBHLQ.BKP.DBASE5 UNIT(Not specified) VOLSER(volOO1)
                   O LIMIT(5)

..
..
```

Figure 12. Example Output from DISPLAY OPTIONS Command Showing Specified LIMIT

The following output from a DISPLAY OPTIONS command shows no LIMIT was specified:

```
19.59.03 IEESYSAS  SOV06900I Parameter file is DSN:SYS1.PARMLIB(SOVPRM00)
Parameter   (O)riginal Parameter file value
            (E)ffective value
            (C)hange value ; in progress
----------- - ------------------------------------------------------
..
BACKDSN     O DBHLQ.BKP.DBASE5 UNIT(Not specified) VOLSER(vol001)
        O LIMIT(Not specified)
..
..
```

Figure 13. Example Output from DISPLAY OPTIONS Command Showing No Specified LIMIT

VOLSER

volser specifies the volume serial number of the volume on which the backup is to reside.

BACKupJouRNal

journal backup file name specifies the backup data set name prefix that is used to backup the database journal described by the **JouRNal** (*journal file name*) parameter. The backup file name prefix can be up to 35 characters long. The database backup is performed using the SOVADMN utility DATABASE BACKUP command. The *journal backup file name* is suffixed during DATABASE BACKUP processing with V*xxxxxxx*, where *xxxxxxx* is the current backup version of the database. This value is the same as the value used to backup the database. The file is allocated to the unit described by the **UNIT** parameter.

This parameter must be specified to perform database restoration from a journal. This parameter is ignored if journaling is not performed.

A message is written during the SOVADMN DATABASE BACKUP processing detailing the number of database backup and journal data sets and those data sets to be deleted.

UNIT

backup unit specifies the destination of the backup files on DASD. SMS might influence the allocation of the journal backup files. *backup unit* should contain only DASD devices, as random access to *journal backup file name* is required during backup processing.

VOLSER

volser specifies the volume serial number of the volume on which the backup is to reside.

DATABASE Keyword Examples

The following are examples of the DATABASE keyword.

Example 1

The following example specifies the VSAM KSDS data set to be used as the database for the ExHPDM server(s). If the database is to be shared among multiple ExHPDM servers, it must be defined to VSAM with SHAREOPTIONS(4,4) specified.

```
DATABASE(EXHPDM.DATABASE)
```

Example 2

The following example specifies an MVS data set name prefix to be used to form the backup data set for this database. The prefix must conform to MVS naming standards and is limited to 35 characters. ExHPDM provides a unique suffix to this string to form the name of the backup data set. This data set is used for the ADMIN DATABASE BACKUP command. Although optional, a database backup data set should be used. The UNIT(SYSALLDA) parameter determines where the data set is to be allocated. Only DASD specifications should be used.

```
BACKUPDSN(EXHPDM.DATABASE.BACKUP UNIT(SYSALLDA))
```

Example 3

The following example specifies the name of the database journal data set. This is a fully qualified data set name. The journal data sets cannot be shared among ExHPDM servers. For example, if there are three ExHPDM servers sharing the same database, three journal data sets must be defined. This definition uses symbolic substitution to derive a unique name based on the ExHPDM server subsystem name (&sovssn) and the current MVS system name (&sysname). The result might be a journal name of'EXHPDM.JRNL.SREM.MVSO'. Although optional, a journal data set should be used.

```
JOURNAL(EXHPDM.JRNL.&SOVSSN..&SYSNAME)
```

Example 4

The following example specifies an MVS data set name prefix to be used to form the backup data set for this journal. The prefix must conform to MVS naming standards and it limited to 35 characters. ExHPDM provides a unique suffix to this string to form the name of the backup data set. Although optional, a database backup journal data set should be used. This definition uses symbolic substitution to derive a unique name based on the ExHPDM server name (&sovssn) and the current MVS system name (&sysname). The result might be a journal name of 'EXHPDM.JRNL.BKUP.SREM.MVSO'.

```
BACKUPJOURNAL(EXHPDM.JRNL.BKUP.&SOVSSN..&SYSNAME UNIT(SYSALLDA))
```

DEVICE Keyword

The DEVICE keyword is an optional keyword that allows a device definition to be defined for any number of stream definitions. The DEVICE keyword ties the device specified in the STREAM DEVICE keyword to specific esoteric or generic names of output stream tape devices. The usage of the device definition is described in “Identify an Input or Output Device to Service the Stream Task” on page 42.

Caution: Device types that are not alike should not be specified or mixed in the same DEVICE definition as only one FLOWLIMIT parameter is allowed for a device definition.

Syntax

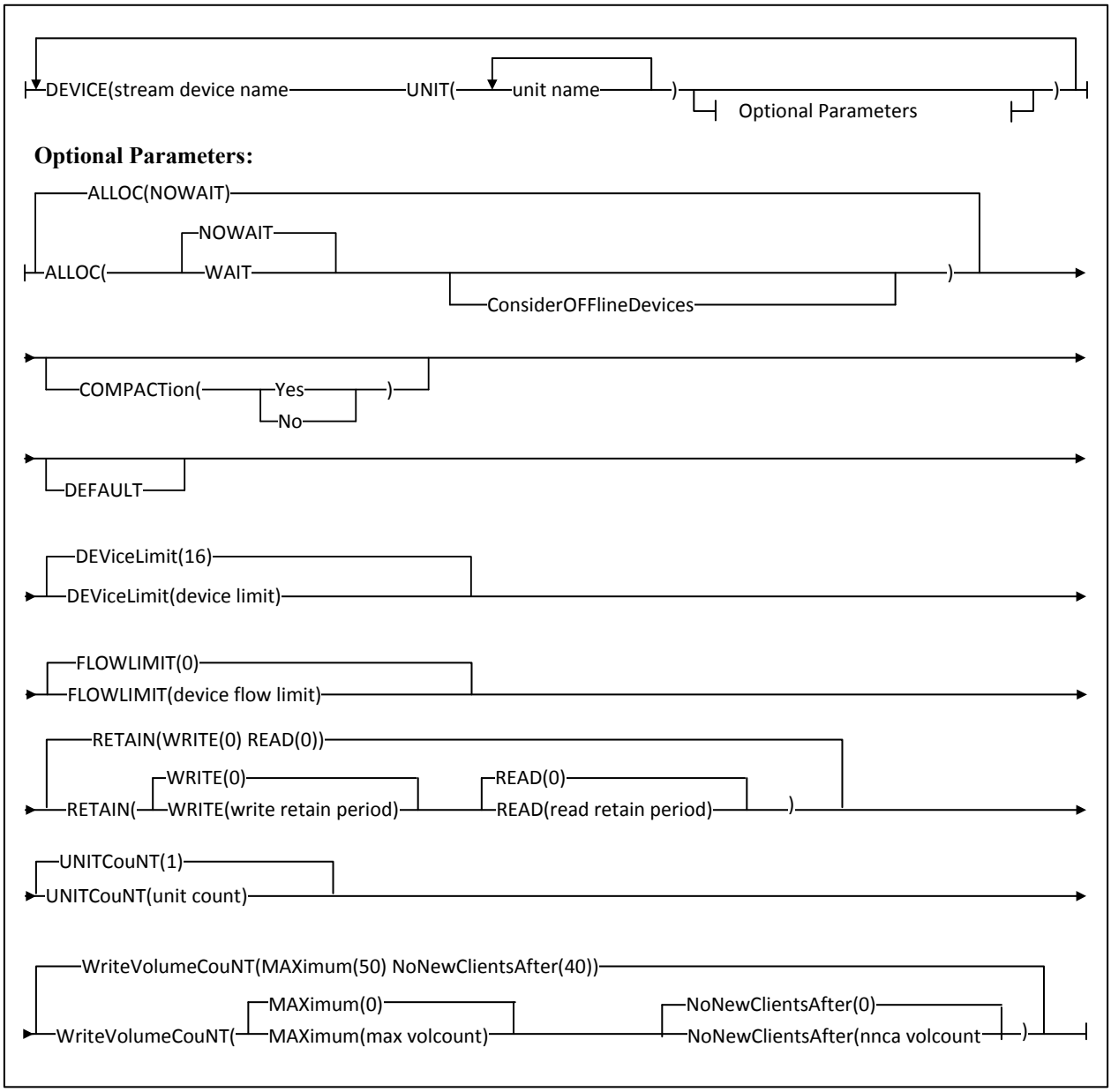


Figure 14. DEVICE Keyword

Keyword Name

DEVICE
initiates the DEVICE keyword and specifies the *stream device name.* The *stream device name* must be a unique name that can be up to eight characters. The first character must be alphabetic.

Parameters

UNIT
unit name specifies a list of esoteric or generic names of the tape devices that may be allocated by a stream task. These devices must be 3490 or 3590 device types. 3480 device types and older are not supported. Each name can be up to eight characters long. Each successive device name in the list is attempted for allocation until an available device is located. Only devices of the same type should be included in the same **DEVICE** definition. The list can contain up to 32 device names.

Where StorageTek's Host Software Component (HSC) is being used to manage tape device allocation, HSC TAPEREQ specifications or HSC SMS esoteric substitution may be used to direct ExHPDM allocations to applicable devices.

Note: Specific MVS device numbers MUST NOT BE USED; use only esoteric or generic device names.

ALLOC

WAIT or **NOWAIT**
specifies that for the tape device allocation, output stream devices listed in the **UNIT** parameter are to wait in allocation retry (WAIT) or not wait (NOWAIT) when the current device name in the **UNIT** list is exhausted. Typically, occurs when an esoteric name is exhausted because all the tape devices are in use. This causes MVS allocation retry to either prompt the operator to respond to the message IEF238D to 'WAIT' or 'CANCEL' or to follow the allocation policy requirements specified in the SYS1.PARMLIB member ALLOC*xx*.

The use of **ALLOC(**WAIT**)** is useful to allow allocations to WAIT when all tape devices are not available. However ALLOC(WAIT) is not required when clients have the WAIT(YES) specification on their DD SUBSYS or the STREAM has WAIT(YES). If these are specified or defaulted, then the client connection waits until the stream task request is redriven by a subsequent connection initiation or a current connection termination. Either of these conditions causes a new stream task to be started and an available tape drive allocated. In a quiet system, when ExHPDM connections are not constantly being performed, the connections can appear to wait indefinitely. To cause stream retry processing automatically the STREAM WAIT(YES RETRYperiod(*time period*)) specification may be used. To cause stream retry processing to be processed manually the **Start Stream** *stream name* operator command may be used.

ALLOC(NOWAIT) is the default.

Note:

1. Care should be taken in choosing ALLOC(WAIT). If no devices are available, then stream startup for all streams requiring devices as specified in UNIT wait until all devices for all streams are available. This can cause outstanding ENQs on SYSZTIOT to hold up all allocation and open/close processing within the ExHPDM server address space. Use of ALLOC(WAIT) and the response to message IEF238D should be reviewed and carefully considered. The recommendation is to use, or default to, ALLOC(NOWAIT).
2. If the ExHPDM server has all of devices allocated in the UNIT specification and ALLOC(WAIT) is specified, then the MVS IEF238D message is not issued. If this occurs, and ExHPDM requires an additional device for stream processing, it prompts the operator with the SOV06223W message.

ConsiderOFFlineDevices

specifies that for the tape device allocation, output stream devices listed in the **UNIT** parameter are to wait in allocation retry when all online devices in the current device name in the **UNIT** list is exhausted. This causes MVS allocation retry to either prompt the operator to respond to the message IEF238D to supply a device name or 'CANCEL' or to follow the allocation policy requirements specified in the SYS1.PARMLIB member ALLOC*xx*.

The use of **ALLOC(**COFFD**)** is useful to allow operators to be prompted for offline devices when all online tape devices are in use. The considerations for this parameter specification are similar to those for ALLOC(WAIT).

The default is for ExHPDM to not consider offline devices.

Note: Care should be taken in choosing ALLOC(COFFD). If no devices are available, then stream startup holds up MVS allocation for the ExHPDM server and other allocations in the MVS image while in allocation retry. Additionally, outstanding ENQs on SYSZTIOT hold up all allocation and open/close processing within the ExHPDM server address space. Usage of ALLOC(COFFD) and the reply to message IEF238D should be reviewed and carefully considered.

COMPACTion

YES

specifies that the data should be compacted or compressed. This parameter performs in the same way as the MVS JCL statement, DCB=TRTCH=COMP.

NO

specifies that the data should not be compacted or compressed. This parameter performs in the same way as the MVS JCL statement, DCB=TRTCH=NOCOMP.

If **COMPACTion** is not specified, the installation default for compaction is used.

DEFAULT
specifies that this is the default device entry. Only one default entry is valid. Only the first default entry specified is honored. If no DEFAULT entry is defined, an internal default of: UNIT(CART) DEVL(16) RETAIN(WRITE(0) READ(0)) is used.

DEViceLimit
device limit specifies the maximum number of devices that may be allocated concurrently for the DEVICE definition. This value should always be specified for ExHPDM to correctly process allocations for the device definition. If this value is larger than the number of devices in the UNIT specification, then ExHPDM outputs warning messages when attempting to start new stream tasks because it cannot allocate these units.

DEViceLimit(0) indicates no limit.

DEViceLimit(16) is the default.

FLOWLIMIT
device flow limit specifies the limiting performance level that devices in this **DEVICE** definition can handle. *device flow limit* can be specified in bytes per second (12000000), KB per second (12000K), MB per second (12M), or GB per second (.012G). FLOWLIMIT is used in conjunction with the **ExpectedFlowIncrease** parameter of the CLASS keyword to provide load balancing.

FLOWLIMIT(0) indicates that there is no flow limit. In this case load balancing is performed using the STREAM CONNECTIONS (*max stream connections*).

FLOWLIMIT(0) is the default.

RETAIN
specifies the time that the device(s) remains allocated to a stream task after all client connections are closed. This may be specified for read or write stream tasks.

The specification of RETAIN can be useful to keep a stream file allocated when more work for the stream is expected. Once the RETAIN time period has elapsed, then the stream file and its associated devices are unallocated. When a write stream file is unallocated, it cannot be added to (i.e., DISP=MOD on stream files is not supported).

To see if a stream has entered the RETAIN period processing, issue the DISPLAY STREAM DETAIL operator command. The output of this command indicates if the stream is in RETAIN processing and the remaining time before the stream file is unallocated. For example, the following DISPLAY STREAM DETAIL output shows the read stream in retain processing for the total period

indicated by the Retain Time (10 minutes), and will terminate when the Remaining time (currently 9 minutes and 11 seconds) completes:

```
SOV06055I Stream : HSCISTRM ID : 99 Device : 0B80 VOL : 542486
File : OWPL.HSCISTRM.GTGFS5A5
Read  Flow Rate : 0 KB/sec Blocks : 3502
Status : Waiting
Elapse Time (HH:MM:SS) : 00:03:50
Retain Time (HH:MM:SS) : 00:10:00 Remaining : 00:09:11
```

If more work for the stream is received before the retain period is reached, the RETAIN period is restarted after all client connections are closed.

Client connections can override the retain period of a device used by a stream task through the use of the RETAIN keyword, specified on the DD SUBSYS JCL parameter. When a client connection specifies a RETAIN time interval of zero (RETAIN(0)) or specifies a time interval greater than the current device retain period, this becomes the current retain period for the stream tasks device. Refer to “DD SUBSYS JCL Parameter Examples” on page 280 for details on specifying RETAIN for a client connection.

Note: In addition to accepting the WRITE and READ retain period keywords, a positional value may be specified. The positional value is equivalent to the WRITE retain period. This approach maintains compatibility with older specifications of RETAIN.
For example, RETAIN(10) is equivalent to RETAIN(WRITE(10))
The positional retain period is ignored when a WRITE(*write retain period*) is also specified.

WRITE

write retain period specifies the time that the device remains allocated to WRITE or OUTPUT stream tasks after all client connections are closed. *write retain period* is specified as a time interval as described by “Specifying a Time Interval or Period” on page 161.

WRITE(0), the default, indicates that no retain takes place.

Note: When a write stream task exceeds the volume count, as specified on the NoNewClientsAfter keyword, the WRITE(*write retain period*) is ignored.

READ

read retain period specifies the time that the device remains allocated to READ or INPUT stream tasks after all client connections are closed. *read retain period* is specified as a time interval as described by “Specifying a Time Interval or Period” on page 161.

READ(0), the default, indicates that no retain takes place.

UNITCouNT

unit count specifies the number of devices, from 1 to 50, to be allocated for each allocation of the device specified in the **UNIT** parameter. This value is used to allocate a number of devices for each stream task to allow the premounting of stream file tape volumes as on JCL // DD UNIT=(uname,ucount). **UNITCNT** should be optimally specified as **UNITCNT(2)** for premounting tape volumes. UNITCNT(1) is the default.

WriteVolumeCouNT

specifies the number of volumes that are used in the allocation and processing of a write stream file. Two parameter values may be specified to provide this volume count to the write stream : MAXimum and NoNewClientsAfter. In either case, a volume count value may be specified as a number between 0 and 255 inclusive. A value of zero or the omission of one of the parameters is interpreted as an instruction to compute a suitable default, based on the value given for the other parameter.

For example,

WVCNT(MAX(35) NNCA(0))

will set the NNCA parameter to 25

WVCNT(MAX(0) NNCA(55))

will set the MAX parameter to 65.

If both parameters are specified as zero or are not supplied, the effective values are MAXimum(50) NoNewClientsAfter(40).

MAXimum

max volcount is the value passed to MVS dynamic allocation as the volume count parameter. Before being used in this way, the max volcount value is rounded up by ExHPDM in accordance with the IBM specification for the volume count allocation parameter. This is done in a similar way that the volume-count value is processed on the JCL DD VOLUME=(,,,volume-count) parameter.

This means that the max volcount is rounded up to 5, if max volcount is less than 5, or the next multiple of 15 plus 5, or 255 (whichever is less). For example, MAXimum(2) is rounded up to 5, MAXimum(6) is rounded up to 20 (15 times 1 plus 5), and MAX(250) is rounded up to 255 (not 260).

The max volcount allocation parameter sets a limit to the number of volume serials which can be used for the one stream file. If a stream task continues writing data such that more than max volcount volumes are required, the stream tasks terminate with an ABEND S837 with a reason code 8. ExHPDM terminates any clients that are still using that stream task.

NoNewClientsAfter

nnca volcount specifies the number of volumes that may be used before ExHPDMdisallows any new clients from connecting to a stream. By default, nnca volcount is set to the larger of max volcount minus 10, or 1.

For example, specifying MAXimum(65) sets a default value of NoNewClientsAfter(55). MAXimum(5) sets a default value of NoNewClientsAfter(1). Note that the rounded-up value ofMAXimum is used when computing this default. For example, MAXimum(6) is rounded up to MAXimum(20), from which NoNewClientsAfter(10) is obtained.

Normally, a nnca volcount value is specified which is less than the max volcount value. If nnca volcount is greater than or equal to max volcount, the nnca volcount value is not effective because the stream abends at the end of max volcount volumes before nnca volcount is reached.

When an nnca volcount value, less than max volcount, is given then the stream task continues writing normally until nnca volcount volumes are completely utilized by the stream file. As soon as the next volume is mounted, the stream file is put in a special state which prevents additional clients from connecting to that stream file. Any clients which were already connected to the stream continue until they complete normally, or max volcount volumes have been used, whichever occurs first.

A stream file which writes to more than nnca volcount volumes can be distinguished in the output from the ExHPDM DISPLAY STREAM DETAIL command. The status of such stream files is marked as 'Volume limit'. A stream in 'Volume limit' does not accept new client connections. Any clients attempting to connect are connected to a new stream file. A stream in 'Volume limit' also does not honor any write RETAIN specification. There is no point in doing so, as no new connections to such a stream file could be made. When a stream file makes a transition to 'Volume limit' state, message SOV06600I is issued to the console and ExHPDM log. The following example assumes that MAXimum(20) and NoNewClientsAfter(10) are in effect:

SOV06600I Stream STRNAME task 51 has reached 11 volumes.
3 current clients have 10 volumes to complete.

This message shows the current number of volumes (nnca volcount plus 1), the number of clients currently connected, and the total number of volumes remaining in which those clients may complete, including the current volume, computed as max volcount minus nnca volcount. The ExHPDM command DISPLAY STREAM DETAIL (message SOV06055I) displays the stream status (described above), as well as the volume count parameters in effect. This shows the current number of volumes written, as well as the (adjusted) values of the NoNewClientsAfter and MAXimum parameters which are in effect.

DEVICE Keyword Examples

The following are examples of the DEVICE keyword.

Example 1

The following example defines a stream device, TLDEV, that uses all tape devices specified by the esoteric name AUSALL. A limit of 16 tape devices are only permitted for all usages of this stream device definition. This example is the default stream device specification.

```
DEVICE(TLDEV UNIT(AUSALL) DEVICELIMIT(16) DEFAULT)
```

Example 2

The following example defines a stream device, RWDEV, that uses all tape devices specified by the esoteric name RWALL. A limit of 2 tape devices are only permitted for all usage of this stream device definition. A maximum volume count of 5 volumes is allowed and no new clients are allowed to connect to stream files that have used more than 2 volumes.

```
DEVICE(RWDEV UNIT(RWALL) DEVICELIMIT(2) WVLCNT(MAX(5) NNCA(2))
```

Example 3

In the following example, the stream device definition, REDDEV, specifies the tape devices to be used. The tape devices are named in the unit parameter as AUSRED and RWALL and are esoteric named devices.

The stream file is allocated to AUSRED first and when exhausted, prompts through MVS allocation retry because ALLOC(WAIT COFFD) was specified. If the device cannot be satisfied, the stream file allocation attempts allocation using the esoteric name RWALL. If allocation still cannot be satisfied, the stream task closes and rejects client connections that have the WAIT(NO) specification on DD SUBSYS or the STREAM parameter.

For all stream allocations, use two tape devices from the unit list (UNITCOUNT(2)). This permits premounting of tape media. Do not permit compaction or compression at the tape device (COMPACT(NO)) and limit tape allocations for all usage of this stream device to 16 tape drives.

The stream file remains allocated to the device for up to 15 minutes of inactivity for write processing and 2 minutes of inactivity for read processing (RETAIN(WRITE(15M) READ(2M))). Lastly, the devices specified in this definition have a maximum throughput of 12 MB per second (FLOWLIMIT(12M)). This is used with any EFI specification on the class statement to cause load balancing. In such cases, a new stream task is started based on limits expressed in the concurrent statements.

```
DEVICE(REDDEV UNIT(AUSRED RWALL) UNITCOUNT(2)
FLOWLIMIT(12M) ALLOC(WAIT COFFD) COMPACT(NO)
DEVICELIMIT(8) RETAIN(WRITE(15M) READ(2M)) )
```

LKEYINFO Keyword

Upon installation, you have a 75-day trial period. During this 75-day period, you must secure a permanent license key. Do not wait until this trial period is over to secure your permanent key. Once this trial period expires you cannot start ExHPDM.

The two types of license keys are:

- Permanent: Enables you to use ExHPDM when StorageTek has received the initial license fee for a product or feature.
- Emergency Software Key: For short term use in emergency situations. It is limited to seven days.

Note: After the trial period expires and a valid license key has not been entered or an emergency license key expires, ExHPDM automatically shuts down. When the permanent key expires, no new output connections are allowed. However, you are still able to perform client restores and administrative functions.

Syntax

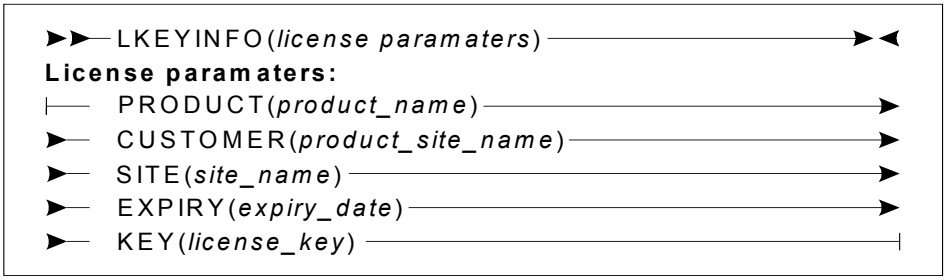


Figure 15. LKEYINFO Keyword

Keyword

LKEYINFO
initiates the **LKEYINFO** keyword.

Parameters

PRODUCT (product_name)
This is the ExHPDM version and release numbers.

CUSTOMER (customer_site_name)
This is your 20-character customer name.

SITE (*site_numbe*r)
This is the site number assigned to you by SMD for your license key and product. The site number consists of four to six numeric characters.

EXPIRY (*expiry_date*)
This is the expiration date in ISO standard YYYYDDD format. Example: 2006009

KEY (*license_key*)

This is your key that is assigned by SMD and is up to 24-characters in length.

LKEYINFO Example

The following is an example of the LKEYINFO keyword.

```
LKEYINFO (CUSTOMER(SMITH)
    SITE (1234)
    EXPIRY (2010365)
    PRODUCT (EXHPDM30)
    KEY (123456789ABCDEF))
```

LOGFILE Keyword

The LOGFILE keyword is a nonstream-related optional keyword for the ExHPDM startup parameter file that defines destination of the server log messages. The SET LOGFILE and SET PRM commands may be used to change the destination of these log messages while the ExHPDM server is active.

Syntax

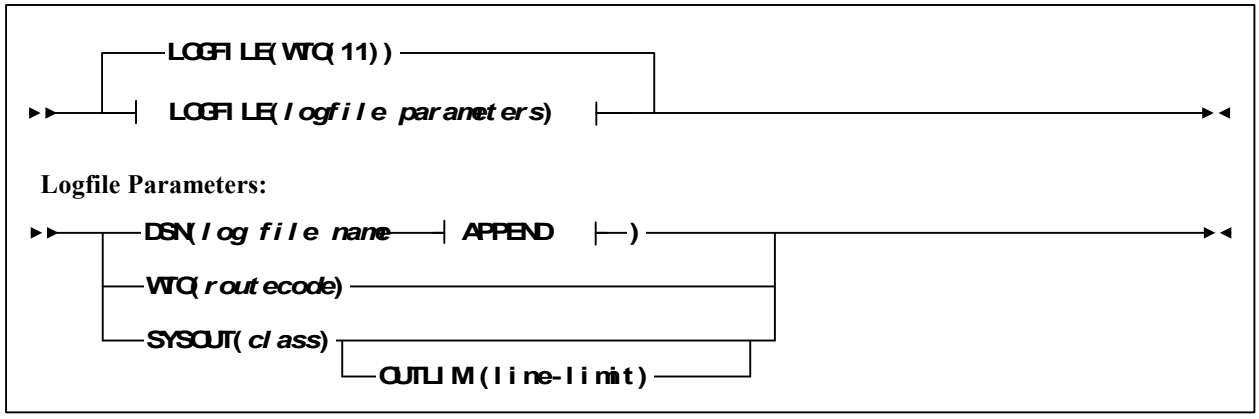


Figure 16. LOGFILE Keyword

Keyword Name

LOGFILE
initiates the LOGFILE keyword.

Parameters

WTO
routecode specifies the console route code. The valid values are 0 to 128. WTO(0) causes the log output to be directed to the MVS SYSLOG.

WTO(11) is the default.

DSN
log file name specifies the MVS data set name. This file is allocated and cataloged if it does not currently exist. If APPEND is not specified and the file exists, the file is over written. If an error occurs while writing to the data set then the log output is redirected to the MVS SYSLOG.

APPEND
specifies that the log file output is to be appended to the file that currently exists and has the same file name.

Note: If the ExHPDM parameter member is shared among multiple ExHPDM servers, the *log file name* must be unique. This file name may be made unique for each ExHPDM server by using system symbols. See *"Sharing Parameter File Definitions Using System Symbols" on page 92*.

SYSOUT

class specifies the JES sysout class.

OUTLIM

specifies the OUTLIM value in the startup parameter file and on the SET command.

sysout_class is any valid JES sysout class.

line_limit indicates the number of output lines that ExHPDM will be allowed to spool to the LOGFILE data set. This line limit value will be used in the dynamic allocation DALOUTLM text unit.

If OUTLIM is set to zero then the installation default might be used. No DALOUTLM value will be set. This is effectively what is being done in ExHPDM V1.1.

If OUTLIM is not specified then the default value will be set to the maximum OUTLIM value of 16777215.

LOGFILE Keyword Example

The following are examples of the LOGFILE keyword.

Example 1

The following example specifies that the server log is to be written to a JES data set (SYSOUT) to class 'E'. A specification to SYSOUT is the preferred way to manage the ExHPDM server log because it can be viewed with any JES spool browser.

```
LOGFILE(SYSOUT(E))
```

Example 2

The following example specifies that messages to be logged from ExHPDM are directed to the MVS data set named EXHPDM.&SYSNAME..&SOVSSN..LOGFILE. Symbolic substitution is used in this parameter to make the log file unique for all ExHPDM servers. See the section *"Sharing Parameter File Definitions Using System Symbols" on page 92* for details about using system symbols.

If the nominated data set does not exist, it is created using the default volume or device for the MVS installation. If the data set fills up (*x*37 ABEND) in the course of ExHPDM logging, further output is directed to the MVS system log.

If APPEND is specified and the ExHPDM.LOGFILE data set already exists, ExHPDM appends to the end of the data set without erasing it.

```
LOGFILE(DSN(EXHPDM.&SYSNAME..&SOVSSN..LOGFILE))
```

Example 3: Output for LOGFILE Parameter Where an OUTLIM Has Been Specified

The following shows possible output displayed for the LOGFILE parameter where an OUTLIM has been specified.

Note: The OUTLIM value is only displayed when SYSOUT is specified for LOGFILE.

```
19.59.03 IEESYSAS  SOV06900I Parameter file is DSN:SYS1.PARMLIB(SOVPRM00)
Parameter   (O)riginal Parameter file value
        (E)ffective value
        (C)hange value ; in progress
------------ - ------------------------------------------------------
..
..
LOGFILE      O SYSOUT(E) OUTLIM(999999)
         E SYSOUT(E) OUTLIM(999999)
..
The following  shows the possible output displayed for the LOGFILE parameter where an OUTLIM(0)
value has been specified :
```

Example 4: Output for LOGFILE Parameter Where an OUTLIM Has Not Been Specified

The following shows the possible output displayed for the LOGFILE parameter where an OUTLIM has not been specified:

```
19.59.03 IEESYSAS  SOV06900I Parameter file is DSN:SYS1.PARMLIB(SOVPRM00)
Parameter   (O)riginal Parameter file value
        (E)ffective value
        (C)hange value ; in progress
------------ - ------------------------------------------------------
..
..
LOGFILE      O SYSOUT(E)
         E SYSOUT(E) OUTLIM(16777215)
..
..
```

Example 5: Output for LOGFILE Parameter Where an OUTLIM (0) Value Has Been Specified

The following shows the possible output for the LOGFILE parameter where an OUTLIM(0) value has been specified:

```
19.59.03 IEESYSAS  SOV06900I Parameter file is DSN:SYS1.PARMLIB(SOVPRM00)
Parameter   (O)riginal Parameter file value
        (E)ffective value
        (C)hange value ; in progress
------------ - ------------------------------------------------------
..
..
LOGFILE      O SYSOUT(E) OUTLIM(0)
         E SYSOUT(E) OUTLIM(installation default)
..
..
```

MANAGEMENT Keyword

The ManaGeMenT keyword is only assigned to client write requests. It is an optional keyword that defines the management attributes of the client data set. The ManaGeMenT is selected by either the SELECT ManaGeMenT, CLASS ManaGeMenT or DD SUBSYS ManaGeMenT keywords as described by “SELECT Keyword” on page 134, “CLASS Keyword” on page 99, and “Both BACKUP and RESTORE steps may require some time to complete. A 300-cylinder database may take between one and 20 minutes to restore, depending on system speed and other workloads. Even starting ExHPDM with a new (empty) database may take a few minutes, since the database needs to be formatted before use.” on page 272. The usage of the management definition is described in “Identify the Stream and Stream Task for the Client Connection” on page 41.

Syntax

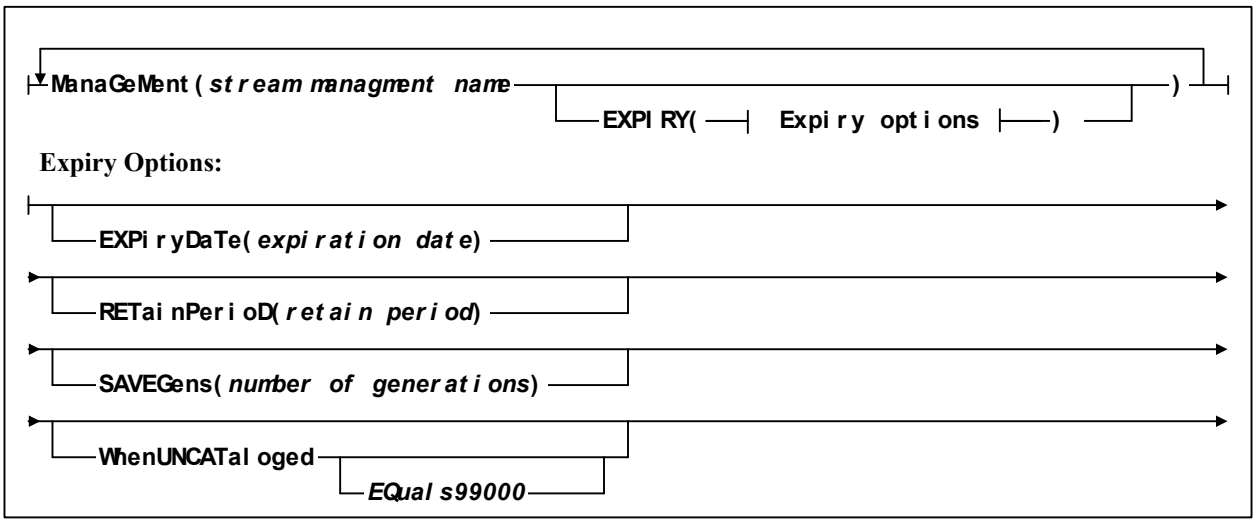


Figure 17. MANAGEMENT Keyword

Keyword Name

ManaGeMenT
initiates the MANAGEMENT keyword.

Parameters

The following parameters apply to the EXPIRY keyword. Each of these indicate under what circumstances the client data set is expired. When a client data set is expired the SOVADMN utility with the ADMIN SCANEXPIRE command is required to delete these from the ExHPDM database.

EXPiryDaTe
expiration date specifies the date at which the client data set is to expire. *expiration date* is specified in the format *yyyy*/*ddd*. Where *yyyy* is a four-digit year and *ddd* is

the three-digit offset from the beginning of but not restricted to that year. For example 2000/900 is converted to 2002/169.

Unlike the JCL EXPDT parameter, the management EXPiryDaTe parameter accepts the expiry date of 1999/365 as a real date. It is not used to denote a never expire condition. EXPiryDaTe(9999/999) could be used to indicate a never expire condition.

If a year or day is specified as 0 (zero), the expiration date is ignore. For example 1999/000.

If the *expiration date* is in the past, the client data is expired immediately.

If a RETainPerioD is also specified, the later of the two dates is used.

RETainPeriod

retain period specifies how long the client data is retained before being expired. *retain period* is specified as a number of days up to 9999.

If an EXPiryDaTe is also specified, the later of the two dates is used.

SAVEGens

number of generations specifies the number of ExHPDM client data set generations that are to be kept for the same client data set name. Older generations rolloff and are expired by ExHPDM when *number of generations* is exceeded. When SAVEGens is specified, this overrides any values specified in RETainPeriod and EXPiryDate.

Note: ExHPDM generations should not be confused with MVS GDGs. ExHPDM generations of a client data set are successive copies of those client data sets that have identical names.

WhenUNCATaloged

specifies that the client data set should be expired by ExHPDM once it becomes uncataloged. If WhenUNCATaloged is specified, this overrides all other expiry values in MANAGEMENT. A usage of this specification is to expire rolled off GDGs.

EQuals99000

when specified, it means that if ExHPDM selects this management class to use for the client dataset and the client DD statement has EXPDT=99000 coded on it, then ExHPDM will treat the client dataset as if WhenUNCATaloged was specified. Normally, when 99000 is coded on a DD statement this overrides any expiry parameter defined in the MANAGEMENT class that ExHPDM selects. ExHPDM would keep the client dataset permanently.

Though it can be used by all clients by using the EQuals99000 parameter on the MANAGEMENT keyword, FDRABR users can specify EXPDT=99000 in the JCL and have ExHPDM manage the client datasets based on whether they are cataloged or not, as would happen if the output

was not being directed to ExHPDM and the TMS used interpreted 99000 as catalog controlled.

MANAGEMENT Keyword Examples

The following are examples of the MANAGEMENT keyword.

Example 1

The following example specifies that the client data sets assigned to management MGMT1 should expire once they become uncataloged.

MANAGEMENT(MGMT1 EXPIRY(WUNCAT))

Example 2

The following example specifies that the client data sets assigned management MGMT2 should expire in 10 days time.

MANAGEMENT(MGMT2 EXPIRY(RETPD(10)))

Example 3

The following example specifies that the client data sets assigned management MGMTOLD should expire at the later time of the expiration date 2010/010 or a retention of 2 years time. RETPD only starts being used when a date 730 days less than 2010/010 is reached.

MANAGEMENT(MGMTOLD EXPIRY(EXPDT(2010/010) RETPD(730)))

Example 4

The following example specifies that only the newest ExHPDM generations of client data sets assigned management SINGLE should be kept. All older versions of the client data set are expired.

MANAGEMENT(SINGLE EXPIRY(SAVEGENS(1)))

MONITOR Keyword

The MONITOR keyword is an optional keyword that specifies the thresholds and intervals for warning messages to be issued about the remaining capacity of the ExHPDM database and journal data sets.

If MONITOR is specified, it is possible to plan for resizing the database before it becomes necessary. Keeping track of the use of the database over time also allows more accurate capacity planning. If MONITOR is not specified, no monitoring is performed.

Caution: Monitoring the database use at excessively frequent intervals degrades system performance because the complete KSDS index set needs to be scanned if the database is shared (SHR(4,4)). You should limit the **INTERVAL** parameter to a minimum of 1 hour in this case.

Because the journal data set is exclusive to each server, monitoring the journal is efficient and is not limited by the restrictions described above.

Syntax

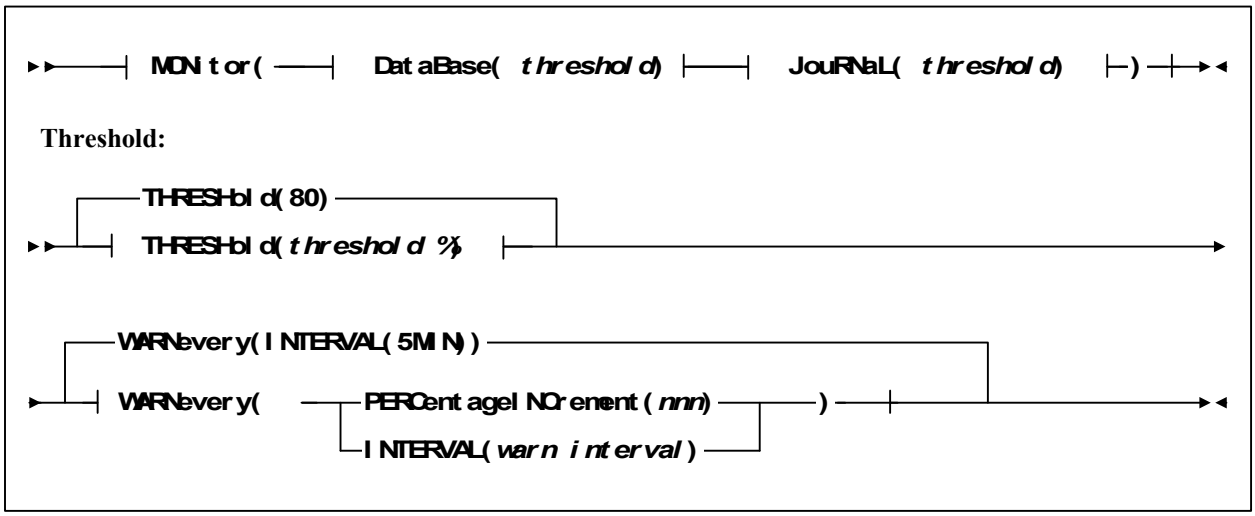


Figure 18. MONITOR Keyword

Keyword Name

MONITOR
initiates the MONITOR keyword.

If the MONITOR keyword is not specified, no monitoring is performed.

If the MONITOR keyword is specified without parameters, both the **DataBase** and **JouRNal** monitoring parameters default to **THRESHold(80)** and **WARN(INTERVAL(5MIN))**.

Parameters

DataBase
specifies the threshold parameters for the DATABASE KSDS. If MONITOR is specified but **DataBase** is not, an initial warning message is issued when the database is 80 percent full and a warning message is issued every 5 minutes. A final message is issued when the database is full. If MON(DB(THRESH(0))) is specified, no monitoring is performed for the database, however the journal data set is monitored as specified by the defaults.

JouRNaL
specifies the threshold parameter for the journal data set. If MONITOR is specified but **JouRNaL** is not, an initial warning message is issued when the journal is 80 percent full and a warning message is issued every five minutes. A final message is issued when the journal is full. If only MON(JRNL(THRESH(0))) is specified, no monitoring is performed for the journal; however, the database is monitored as specified by the defaults.

Thresh for the **DataBase** and **JouRNaL** parameters contains the following subparameters.

THRESHold
threshold% specifies how full the ExHPDM database or journal data set must be before the ExHPDM server begins issuing warning messages. *threshold%* is specified as a decimal number from 0 through 100. If 0 is specified, no monitoring is performed and any values for **WARNevery, PERCentageINCrement,** and **INTERVAL** are ignored.

THRESHold(80) is the default.

WARNevery
specifies the frequency at which subsequent warning messages are issued after the initial warning message at the percentage specified by **THRESHold**. If **THRESHold(0)** is specified, only one warning message is issued and subsequent parameters are ignored. When **WARNevery** is specified, either **INTERVAL** or **PERCentageINCrement** must also be specified.

PERCentageINCrement
increment% specifies the amount of incremental use that must occur in the database or journal data set before the next warning message is issued. *increment%* is specified as a decimal from 0 through 100. Warning messages are triggered when this incremental increase is met. This parameter is ignored if THRESHold(0) is specified.

INTERVAL
warn interval specifies a decimal number, greater than or equal to 1, that is the minimum time interval between repeated warning messages. *warn interval* is specified as a time interval as described by “Specifying a Time Interval or Period” on page 161. This parameter is ignored if THRESHold(0) is specified.

The database will only be monitored when an update occurs from the server that is performing the monitoring. Therefore, the ExHPDM server, which is generating the

majority of database update activity (that is, creating the most stream files), should be the server that is selected for the MONITOR keyword because this gives the most timely warnings.

If the **INTERVAL** parameter is set to 2 minutes, for example, but the MONITOR server is only performing one update every 10 minutes, the actual monitor message interval is 10 minutes. This is the case, even if other servers which share the database are generating updates at a higher frequency.

The actual percentage of use figures reported by the MONITOR server depends on the VSAM share options specified for the database. The rules are:

SHR(1,3)—non–shared
Percentage use is calculated using the AVSPAC and HALCRBA fields obtained from the VSAM ACB. The sequence set is not scanned. This effectively gives the percentage of available bytes.

SHR(3,3)—cross–region shared
Percentage use is obtained based on the number of free CIs in the KSDS. Sequence set is not scanned.

SHR(4,4)—cross–system shared
Percentage use is based on the worst case free CIs in any CA. The index sequence set needs to be scanned to perform this calculation. For each CA in the KSDS, the corresponding index record for that CA is read. That is why, for SHR(4,4), the KSDS should be allocated in units of cylinder; this provides the largest number of CIs per CA and gives the best resolution and efficiency. Note that if the KSDS is allocated with a small number of CIs per CA (for example, 6), the percentage of use can only be quoted as 0, 16, 33, 50, 67, 85, or 100 percent.

MONITOR Keyword Example

The following is an example of the MONITOR keyword.

Example 1

The following example indicates that database monitoring should start displaying warning messages once the 70 percent threshold is reached. The warning message are issued when the database increases in size by 5 percent increments. Journal monitoring starts displaying warning messages at the 85 percent threshold and a warning is again displayed at a minimum interval of 2 minutes.

```
MONITOR(DATABASE(THRESHOLD(70) WARNEVERY(PERCINC(5)))
  JOURNAL(THRESHOLD(85) WARNEVERY(INTERVAL(2MIN))) )
```

Example 2:

The following example shows messages issued during database and journal formatting:

Database Formatting Messages

Message SOV09xxxI Database %s is being %s. Please wait...

Where the fields are defined as :

- Name of database data set name
- Character string to indicate action, eg. 'Allocated', 'Initialized', or 'Formatted',

Once the formatting has completed the following messages will be issued.

Message SOV09xxxI Database %s %s complete.

Where the fields are defined as :

- Name of database data set name
- Character string to indicate action, eg. 'Allocation', Initialization', or Formatting'.

For example, the following messages will be issued when database formatting is required.

Message SOV09xxxI Database DBHLQ.DBASE is being formatted. Please wait...
Message SOV09xxxI Database DBHLQ.DBASE formatting complete.

Journal Formatting Messages

Message SOV09xxxI Journal %s is being %s. Please wait...

Where the fields are defined as :

- Name of journal data set name
- Character string to indicate action, eg. 'Allocated', 'Initialized', or 'Formatted',

Once the formatting has completed the following messages will be issued.

Message SOV09xxxI Journal %s %s complete.

Where the fields are defined as :

- Name of journal data set name
- Character string to indicate action, eg. 'Allocation', Initialization', or Formatting'.

For example, the following messages will be issued during journal dataset allocation.

Message SOV09xxxI Journal DBHLQ.JOURNAL is being allocated. Please wait...
Message SOV09xxxI Journal DBHLQ.JOURNAL allocation complete.

PREFIX Keyword

The PREFIX keyword is a non-stream related optional keyword for the ExHPDM startup parameter file. The PREFIX keyword is used to specify either or both prefixes for ExHPDM messages and operator commands.

Syntax

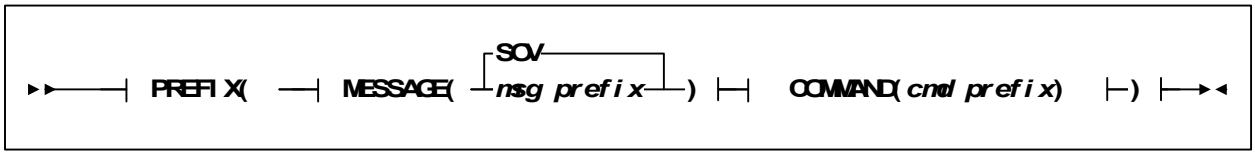


Figure 19. PREFIX Keyword

Keyword Name

PREFIX
initiates the PREFIX keyword.

Parameters

MESSAGE
msg prefix specifies the three alphanumeric characters to prefix ExHPDM message numbers.

MESSAGE(SOV) is the default.

COMMAND
cmd prefix specifies the prefix to the ExHPDM operator commands. This parameter can be a maximum of 15 characters. ExHPDM only recognizes commands that are entered with this command prefix value. If special (non-alphanumeric) characters are included in the command prefix, single quotes must be placed around the prefix.

Care should be exercised to avoid the duplication of valid commands of other MVS components and operating system software. However, ExHPDM does not prevent the command from processing but also responds to the command.

This parameter defaults to the ExHPDM subsystem name; that is, COMMAND(&SOVSSN). &SOVSSN is an ExHPDM supplied substitution variable as described by *"Sharing Parameter File Definitions Using System Symbols" on page 92*.

Note: If there are multiple servers on the same MVS image using the same command prefix, all of the servers respond when a command is entered. This does not occur if the default command prefix is used, as each server must have a unique MVS subsystem name.

PREFIX Keyword Examples

The following are examples of the PREFIX keyword.

Example 1

The following example specifies the character string to be used for ExHPDM message prefixes and the ExHPDM operator command string. The MESSAGE prefix is set to ‘ABC’ and the COMMAND prefix is to ‘EXHPDM’.

```
PREFIX(MESSAGE(ABC) COMMAND(EXHPDM) )
```

Example 2

The following example uses symbolic substitution to define the character string to be used for ExHPDM message prefixes and the ExHPDM operator command string.

Symbolic substitution is used to form a MESSAGE prefix from &SYSCLONE and the last character of the ExHPDM server’s MVS subsystem name using &SOVSSN(–1:1). For example, if &SYSCLONE is ’VO’ and &SOVSSN is ’SREM’, the message prefix is ’VOM’.

Symbolic substitution is used to form an ExHPDM command prefix from the string ‘HPDM’ and the ExHPDM server name in &SOVSSN. For example, if &SOVSSN is ‘SREM’ then the command prefix is ‘HPDMSREM’.

```
PREFIX(MESSAGE(&SYSCLONE&SOVSSN(-1:1))
COMMAND(HPDM&SOVSSN) )
```

REQUEST Keyword

The REQUEST keyword is a non-stream related optional keyword that sets the limits of the resources for the ExHPDM server. The values set in the REQUEST keyword cannot be dynamically refreshed using the SET PRM command. The ExHPDM server must be restarted to use altered values set in SOVPRM*xx*.

Syntax

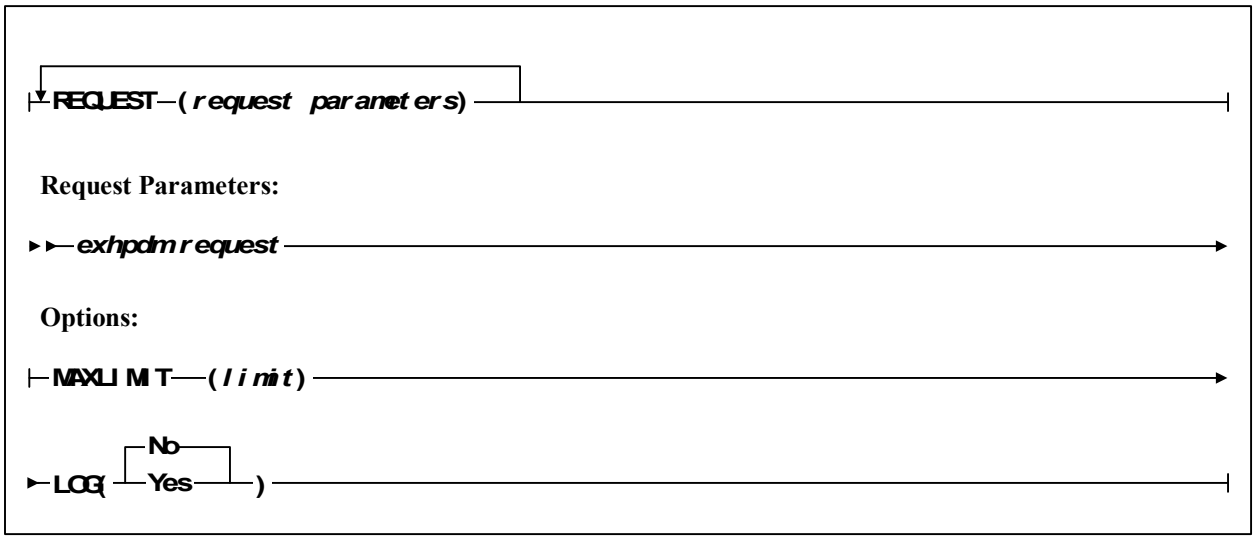


Figure 20. REQUEST Keyword

Keyword Name

REQUEST
initiates the REQUEST keyword.

Parameters

ADMIN
specifies ADMIN requests made through SOVADMN.

OPERCMD
specifies operator command requests.

Caution: If **OPERCMD** is specified and **MAXLIMIT** is set to zero, the operator cannot communicate with ExHPDM.

CONNECT
specifies client connection requests.

VALIDATE
specifies validation requests. These are performed during DDNAME allocation to ensure that a client connection is correct. That is, the DD SUBSYS parameters are valid.

MAXLIMIT

Caution: If **OPERCMD** is specified and **MAXLIMIT** is set to zero, the operator cannot communicate with ExHPDM.

The default and maximum values for **MAXLIMIT** are:

Table 2. MAXLIMIT Default and Maximum Values

Request	Default	Maximum Value
ADMIN	20	200
OPERCMD	20	200
CONNECT	200	5000
VALIDATE	200	1000

When the number of requests, specified by the **MAXLIMIT** parameter, is reached, ExHPDM rejects any further requests. In addition, the **MAXLIMIT** parameter values can only be changed when ExHPDM is restarted. That is, a SET **PRM** command does not change the effective values. The DISPLAY **OPTIONS** operator command shows the parameter settings for **MAXLIMIT**.

LOG

specifies whether (YES) or not (NO) the messages generated by the request should be directed to the ExHPDM log. This parameter works much the same way as the **LOG** parameter on the STREAM keyword and the DD SUBSYS JCL parameters. The values for the **LOG** parameter specified on the STREAM keyword and the DD SUBSYS JCL parameters override the REQUEST **LOG** value.

LOG(NO) is the default.

REQUEST Keyword Examples

The following are examples of the REQUEST keyword.

Example 1

The following example specifies that a maximum of 10 SOVADMN directives can be active simultaneously. The system print output from the ADMIN utility is also to be directed to the ExHPDM server log.

```
REQUEST(ADMIN MAXLIMIT(10) LOG(YES))
```

Example 2

The following example specifies that a maximum of 200 client connections are to be permitted. The statistics and messages for each connect request are also to be directed to the ExHPDM server log.

```
REQUEST(CONNECT MAXLIMIT(200) LOG(YES))
```

Example 3

The following example specifies that a maximum of 5 ExHPDM operator commands can be active simultaneously and 10 connection requests. The operator command results and ADMIN output are to be directed to the ExHPDM server log.

```
REQUEST(OPERCMD MAXLIMIT(5) LOG(YES))
REQUEST(CONNECT MAXLIMIT(10))
REQUEST(ADMIN LOG(YES))
```

SAF Keyword

The SAF keyword is a non-stream related optional keyword for the ExHPDM startup parameter file.

Syntax

Figure 21. SAF Keyword

Keyword Name

SAF
initiates the SAF keyword.

Parameters

CLASS
saf class name specifies a user-defined SAF resource profile name used for ExHPDM resource verification. See “System Security, Authorization, and Verification” on page 46 for details on the security aspects of ExHPDM and the SAF profiles that may be used to control access to ExHPDM resources.

CLASS(FACILITY) is the default.

SAF Keyword Example

The following example specifies a user-defined MVS SAF resource class to be used for ExHPDM stream class and stream name validation.

Note that the definition of a specialized SAF class is a matter of security integrity. In this example, SAF(CLASS($EXHPDM)), the $EXHPDM class is used for resource validation. If a non–IBM SAF resource class is required, it needs to be implemented by the system programming and security personnel. Such a change requires a system IPL. Refer to the appropriate RACF publication.

```
SAF(CLASS($EXHPDM))
```

SELECT Keyword

The SELECT keyword is only used for client write requests. The SELECT keyword is an optional keyword that identifies to ExHPDM which client connections belong to which stream class definition. The rules are searched in the sequence that they are defined in the parameter library. The first matching rule entry is the one that is used. The **EXCLUDE** parameter can be specified instead of the **CLASS** parameter to indicate that a particular connection should not be under the control of ExHPDM. The usage of the SELECT keyword is described in “Identify a Client Connection for ExHPDM Processing” on page 36.

Syntax

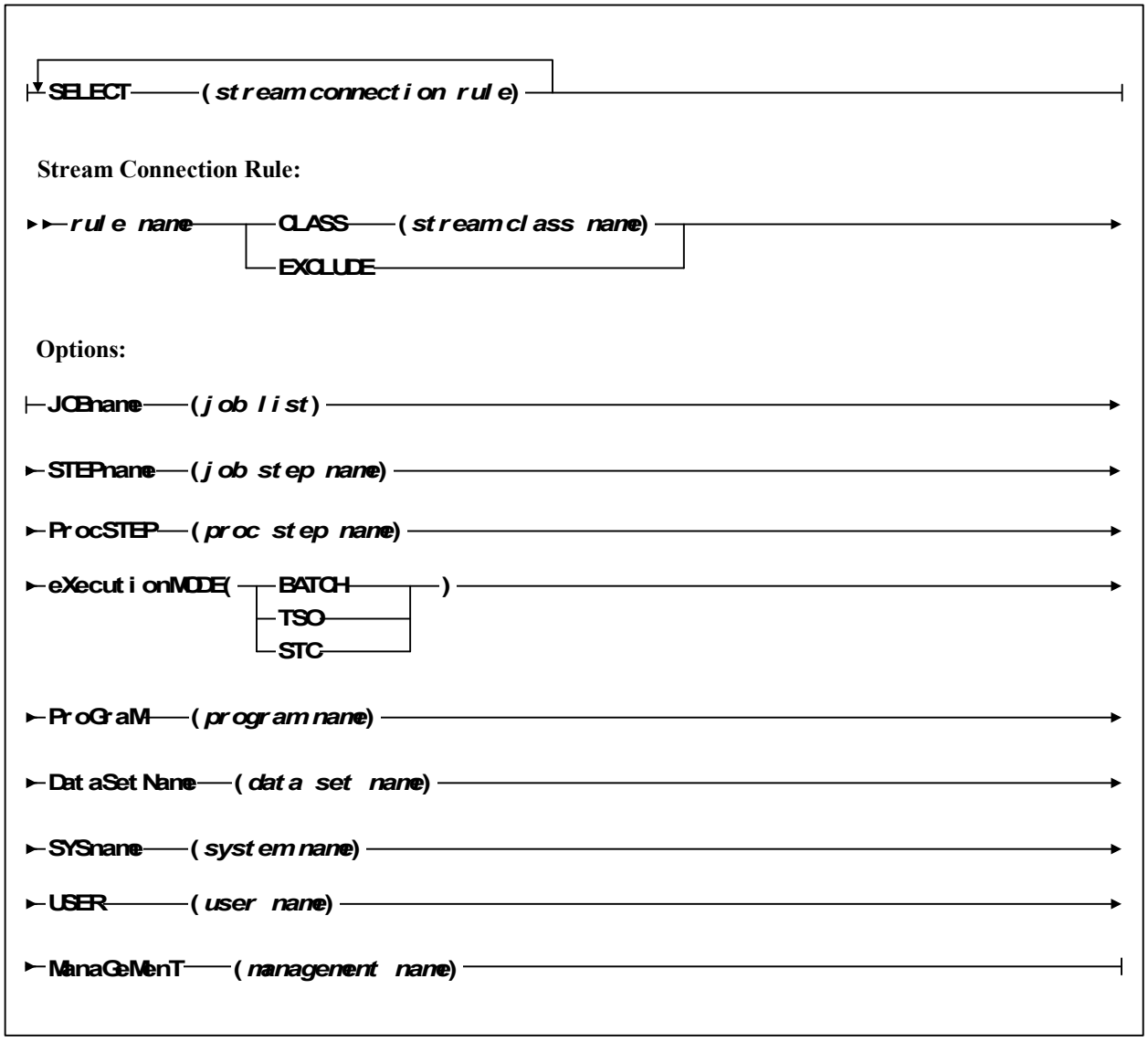


Figure 22. SELECT Keyword

Keyword Name

SELECT
initiates the SELECT keyword.

Parameters

CLASS
stream class name specifies the class name to assign to client connections that match this selection rule.

EXCLUDE
specifies that client connections that match this selection rule are not allowed to connect to ExHPDM. Their connection is rejected.

ManaGeMenT

stream management name specifies the management name to assign to client connections that match this selection rule. This value is not used if a ManaGeMent is specified on DD SUBSYS parameters or an EXPDT or RETPD was specified on the client connection JCL.

The Table 3 (below) shows the rule selection keywords that can apply to a rule. A single match on a list entry in a particular keyword (OR condition) in addition to a match on all specified keywords (AND condition) is required for a rule to be accepted. If a keyword is not specified in the selection rule, it is not used in this matching process.

Table 3. Rule Selection Keywords

Rule Keyword	Max Characters	Values	Description
JOBname	8	Maskable List. See Table 4.	Match to current client name. JOBname is used according to the type of client performing a connection: TSO: userid STC: Started procedure name (S JOB) JOB: Job identifier from JOB card The JOBname is equivalent to the jobname (jjj) or userid (TSO) displayed by the MVS D A,L command as described by message IEE114I.
STEPname	8	Maskable List. See Table 4.	Match to current client step name. STEPname is used according to the type of client performing a connection: TSO: N/A STC: Started task job identifier (S JOB.STEP) JOB: Current job step from EXEC card or the stepname for a step that called a catalogued PROC. The STEPNAME is equivalent to the stepname (sss) displayed by the MVS D A,L command as described by message IEE114I.
ProcSTEP	8	Maskable List. See Table 4.	Match to current client proc step name. procSTEP is used according to the type of client performing a connection: TSO: N/A STC: Current job step from EXEC card JOB: Current job step from EXEC card within a PROC. That is, the step name within a PROC that was called by the step in STEPname. Blank if there is no procedure or procedure step name. The procSTEP is equivalent to the step name within a procedure (ppp) displayed by the MVS D A,L command as described by message IEE114I.
eXecutionMODE	n/a	BATCH,TSO, or STC	Match to execution,mode of the client. BATCH: Batch jobs TSO: TSO address spaces STC: Started tasks
ProGraM	n/a	Maskable List. See Table 4.	List of program names to match the program name of the client.
DataSetName	44	Maskable List. See Table 4.	Match to current data set name.

Table 3. Rule Selection Keywords

Rule Keyword	Max Characters	Values	Description
SYSname	8	Maskable List. See Table 4.	Match to current client system name (CVTSNAM).
USER	8	Maskable List. See Table 4.	Match to current client user ID. Ignored if security is not installed.

The following table shows the simple masking characters that can be used in the rule keywords.

Table 4. ExHPDM Masking Characters

Masking Character	Description
*	Match on any number of characters (including 0) **Example:** A* will match to AB, A, ABCD will not match ZB *A* will match to AB, ZAB, ZA will not match ZB
**	Match on any number of characters across any number of levels (including 0). Intended for data set name matching. **Example:** A.** will match to A, A.B. will not match to B.A.C **.A.** will match to B.A.C, A, B.A A.**.C will match to A.C, A.B.C will not match to A.B
%	Match on any single character. **Example**: A% will match to AB, AC will not match to A, B

SELECT Keyword Examples

The following are examples of the SELECT keyword.

Example 1

The following example describes the SELECTION rules for a series of MAYNARD.BTK data sets. Note that the most specific rules are listed first. Various CLASS definitions are selected based on these names.

```
SELECT(TLRULE1 CLASS(LANBACK)
DATASETNAME(MAYNARD.BTK.FULL.*.SEC1) )
SELECT(TLRULE2 CLASS(DB2PROD)
DATASETNAME(MAYNARD.BTK.FULL.*.SEC2) )
```

```
SELECT(REDRU3A CLASS(IMSPROD)
DATASETNAME(MAYNARD.BTK.RD3A.FULL.*) )
SELECT(REDRU3B CLASS(REDCL3B)
DATASETNAME(MAYNARD.BTK.RD3B.FULL.*) )
SELECT(REDRU3C CLASS(SEISMIC)
DATASETNAME(MAYNARD.BTK.RD3C.FULL.*) )
SELECT(TLRULE CLASS(OPERPROD)
DATASETNAME(MAYNARD.BTK.**) )
```

Example 2

The following example describes the route connection to class DBAPROD when the client job name matches DBAPROD*.

```
SELECT(TLDBA CLASS(DBAPROD) JOB(DBAPROD*) )
```

Example 3

The following example describes the route connection to class IMSPROD when the client jobname matches RQ74*.

```
SELECT(RED3JOB CLASS(IMSPROD) JOB(RQ74*) )
```

Example 4

The following example describes the route connection to class IMSPROD when the client jobstep matches DBAP*.

```
SELECT(RED3STEP CLASS(IMSPROD) STEP(DBAP*) )
```

Example 5

The following example describes the route connection to class IMSPROD when the client procstep matches CGA* or A9X*

```
SELECT(RED3PROC CLASS(IMSPROD) PSTEP(CGA* A9X*) )
```

Example 6

The following example describes the route to any connection when an owner of MAYNARD with a system name of ECC10 and a data set name that matches SEISMIC.WEEKLY.BACKUP.* to a stream class named IMSPROD.

```
SELECT(SEISMIC CLASS(IMSPROD) USER(MAYNARD) SYS(ECC10)
DSN(SEISMIC.WEEKLY.BACKUP.*)  )
```

Example 7

The following example does not permit any connections to this ExHPDM server whose jobname begins with SATT from users OWPL and OWSH.

```
SELECT(PRODSATT EXCLUDE JOBNAME(SATT*) USER(OWPL OWSH))
```

SMF Keyword

The SMF keyword generates SMF records for the purposes of transaction auditing, performance monitoring, reporting, and customer billing. Additional SMF data is produced for integrating FujiSoftek's DR Manager and 21st Century's DR/VFI products for MVS.

SMF record generation is specified in the ExHPDM parameter file SOVPR*Mxx* with the following syntax. The usage of SMF in SOVPR*Mxx* is optional. If it is not specified then no SMF recording will take place:

Syntax

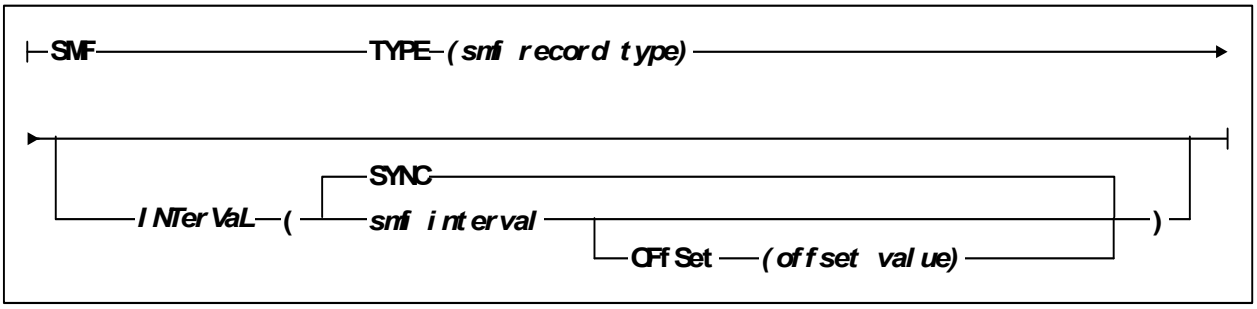


Figure 23. SMF Keyword Syntax Diagram

Keyword Name

SMF
initiates the SMF keyword.

Parameters

TYPE
smf_record_type is an integer between 128 and 255 describing the SMF record type to be used by ExHPDM. ExHPDM uses subtypes of this record type to distinguish different kinds of information.

INTerVaL
smf_interval is a time period describing the interval between writing interval records. Zero (0) means no interval records are to be generated. This is an optional parameter, and if not specified then no interval records are generated. Any interval time format described in the user documentation may be used (see "Specifying a Time Interval or Period" on page 161).

OFfSet
offset_value is a time period describing an offset in the hour to synchronize a specified *smf_interva*l. For example, if *smf_interval* is specified to be 30 minutes, and *sync_value* is specified as 15 minutes, SMF interval records are produced every 15 minutes and 45 minutes past the hour. *offset_value* is an optional parameter, and can only be specified if *smf_interval* is specified.

SYNC

specifies that ExHPDM SMF interval records should be produced using the global SMF interval parameters (specified in the SMFPRMxx member). If no option for INTerVaL is specified, SYNC is the default.

The SMF parameter is optional. If the SMF parameter is not specified, then no SMF records will be generated.

SMF Keyword Examples

Example 1

Write SMF interval records of type 195 at 60 minute intervals 15minutes past every hour

```
SMF(TYPE(195) INTV(30m OFS(15))
```

Example 2

SMFPRMxx Specification for Example 1

```
.....
TYPE(...,195,...)
INTVAL(30)
.....
```

Note: References to record subtypes apply to ExHPDM subtypes and not SMFPRMxx–defined subtypes.

STREAM Keyword

The STREAM keyword is an optional keyword that specifies to ExHPDM how the particular client connections are streamed to a particular device definition. Streams are used to group types of work. The usage of the STREAM definitions is described in “Identify the Stream and Stream Task for the Client Connection” on page 41.

Syntax

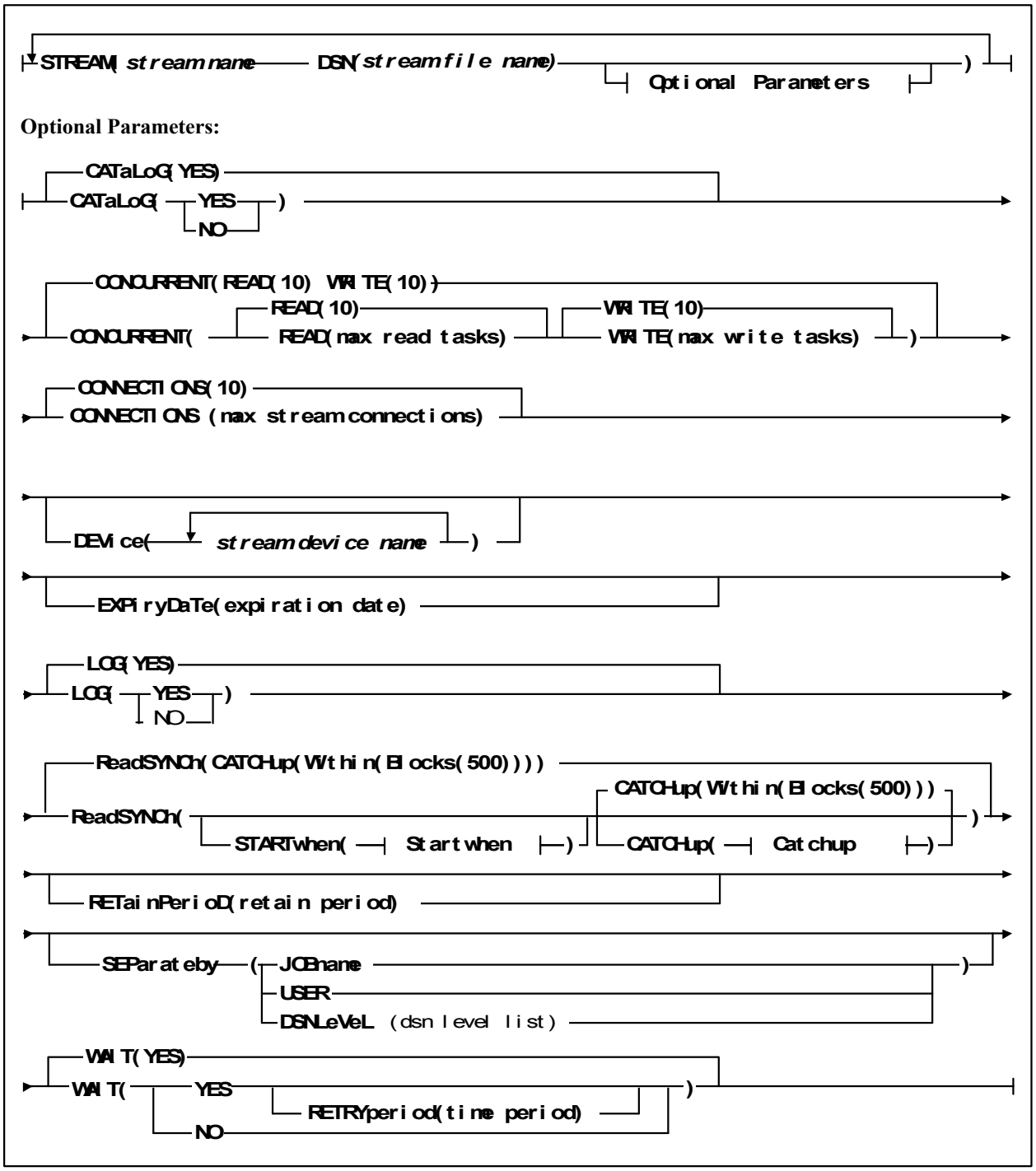


Figure 24. STREAM Keyword

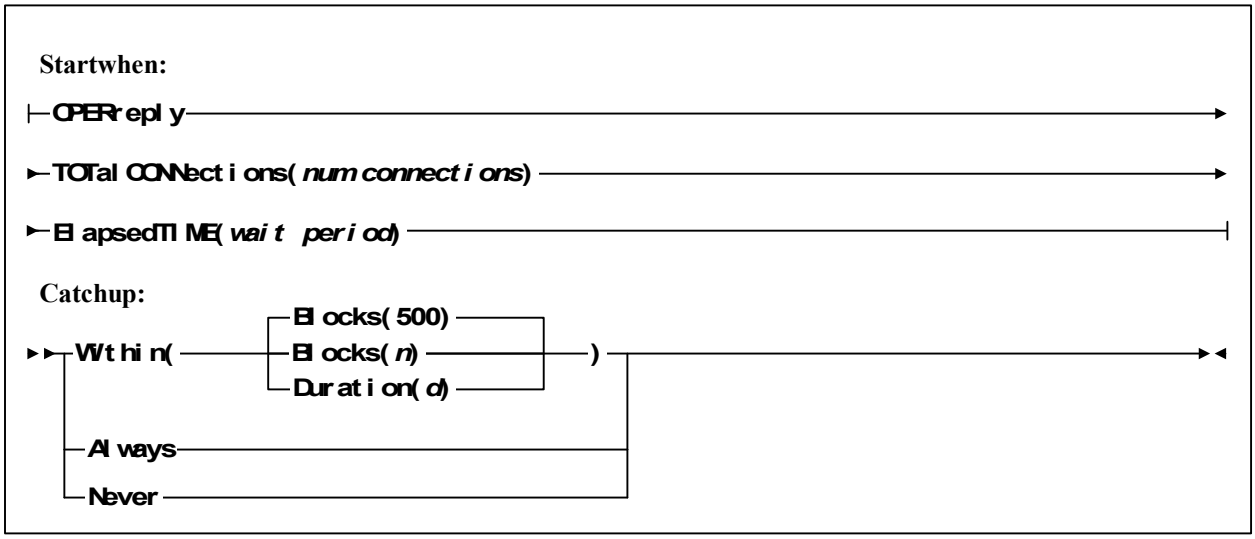


Figure 25. Readsynch Options

Keyword Name

STREAM
initiates the STREAM keyword.

Parameters

stream name
specifies a unique stream definition name. This name can be a maximum of 8 characters and must begin with an alphabetic character.

DSN
stream file name specifies the MVS data set name prefix used to store the data from clients for this stream. The data set name prefix can be a maximum of 35 characters. All *stream file name*s are ICF cataloged unless **CATaLoG(NO)** is specified.

An 8-character suffix (based on the time–date stamp) is formed to make the data set name unique.

Symbolic substitution can be used in the *stream file name* so it contains meaningful information at the time that the stream file is created. This is achieved by using system symbols described in the section *"Sharing Parameter File Definitions Using System Symbols" on page 92* and using double & (i.e., &&) to prefix the symbol name. For example,

```
DSN(ExHPDM.T&&LHHMMSS..STRM01)
```

&LHHMMSS is a standard system dynamic symbol for local time. By using &&LHHMMSS, the symbolic substitution is not performed directly for the local time at ExHPDM startup or by SET PRM processing, but is performed whenever a new stream file is created. Any system symbols can be used in the stream file name as long as the resolved name is not greater than 35 characters.

CATaLoG

specifies whether the stream file is to be cataloged (YES) or not cataloged (NO).

Note: If CATaLoG(YES) is specified or defaulted for a multiple volume stream file and CA–1 is the tape management system, CA–1 chains the volumes in its TMC to match the stream file catalog entry. However, if ExHPDM is defined to CA–1 as an External Data Manager (EDM), CA–1 does not chain the volumes and issues messages TMSSMF13W and TMSSMF30I. This is just a warning and there are no errors or problems caused with the tape management system. To avoid these messages, specify CATaLoG(NO) when using CA–1 and ExHPDM as an EDM.

CATALOG(YES) is the default.

CONCURRENT

READ

max read tasks specifies the maximum number of read stream tasks that can be concurrently active for the STREAM *stream name*. A new read stream task is started for each unique stream file that is requested by client connections.

READ(10) is the default.

WRITE

max write tasks specifies the maximum number of write stream tasks that can be concurrently active for the STREAM *stream name*. A new write stream task is started when the accumulated expected performance levels specified by the CLASS ExpectedFlowIncrease parameter exceeds the limit specified by the DEVICE FLOWLIMIT parameter or the maximum number of connections is reached as specified by STREAM CONNECTIONS parameter.

WRITE(10) is the default.

CONNECTIONS

max stream connections specifies the limit of the number of parallel connections that can be made to a stream task. When this limit is reached, another stream task is started for the *stream name* up to the limit of stream tasks imposed by the STREAM CONCURRENT parameter.

If CONNECTIONS(0) is specified the number of connections is determined by the performance limitations of the selected DEVICE as specified by its FLOWLIMIT parameter and the ExpectedFlowIncrease parameter of the CLASS keyword.

CONNECTIONS(10) is the default for write processing. The maximum number that can be specified is 100. The connection value for read processing is always 100.

DEVice

stream device name specifies the DEVICE definition that may be used by the stream. A list of *stream device name* may be supplied. Each entry is separated by space(s) or a comma. The first DEVICE entry that has not exceeded the limit specified by the DEViceLimit parameter of the DEVICE keyword is used by a stream task. If DEVICE is not specified, the default DEVICE definition is used.

EXPiryDaTe

expiration date specifies the expiry date of the stream file. The date is specified as *yyyy*/*ddd* where *yyyy* is the year and *ddd* is the Julian day. Non–DATE specifications, for example 1998/000, can be specified. The default is no expiration date. EXPDT is used by MVS allocation when allocating the stream file and not further used by ExHPDM; namely it will not appear as the stream expiry date in a ADMIN DataBase LIST and it is not used by ADMIN SCANEXPIRE processing.

Since ExHPDM allows a more flexible format for entering expiry dates, such as day number greater than 366, ExHPDM modifies the given expiry date to conform to JCL conventions. The IBM JCL rules for specifying an expiry date require:

- The year is in the inclusive range 1900 to 2155.
- The day is in the inclusive range 0 to 366.
- Day number 366 is only allowed when the year is a leap year, or the year is 1999.

When an expiry date is specified in the STREAM definition, it is converted to a JCL–compatible date as follows:

- If the year is 1999 and the day is 366, this is passed through unchanged.
- If the day number is greater than the number of days in the given year, the year is incremented by 1 and the day decremented by the number of days in the year. This is repeated until a valid date is obtained.
- If the year is less than 1900, it is made equal to 1900.
- If the year is greater than 2155, it is made equal to 2155.

Warning: EXPiryDaTe should only be used for TMS triggers and **NOT** to expire the stream file and its associated volumes. The expiry date is passed directly to MVS allocation processing and is not used by ExHPDM in any of its own expiry processing. ExHPDM expiration is performed using the client JCL EXPDT, RETPD or MANAGEMENT specifications and the STREAM RETentionPerioD specifications. For further details on TMS triggers see “Using Expiration Date Processing triggers” on page 52.

LOG

specifies whether or not it is required to log ExHPDM messages for client connections messages to the ExHPDM LOGFILE. These are the messages that are also sent to the client JESYSMSG file.

LOG(YES) is the default.

ReadSYNCh

The ReadSYNCh keyword is used to allow synchronization of client read processing either before (STARTwhen) or after (CATCHup) the stream task starts reading a stream file. This parameter is ignored for write clients.

STARTwhen

STARTwhen specifies when a read stream file is to start reading blocks from the tape, given that at least one client is ready to receive data. This keyword allows one or more subparameters to be specified. Each subparameter specifies a particular condition which needs to be met before the stream starts reading blocks from the tape. If more than one condition is specified, the first condition which is satisfied will cause the read process to start. If no conditions are specified, then the read process will start as soon as at least one client is ready to receive data.

Note: The use of STARTwhen will prevent stream task processing from commencing until one of the specified conditions has been satisfied. It is therefore advisable to use more than one STARTwhen condition to prevent stream tasks from appearing to never start. This can occur, for example, when TOTalCONNections is set to a value that is never reached. Under these circumstances the additional specification of OPERreply and/or ElapsedTIME can allow the stream to start.

ElapsedTIME

This condition is satisfied when the elapsed time, since the first client connection, is greater than *time period*. *time period* is specified as a time interval as described by “Specifying a Time Interval or Period” on page 161. If *time period* is 0, this condition is ignored.

TOTalCONNections

This condition is satisfied when the total number of clients connected to this stream is equal to or greater than *num connections*. If *num connections* is 0, this condition is ignored.

OPERreply

This condition is satisfied when the operator replies 'U' to the console prompt messages SOV06500I and SOV06501I.

CATCHup

Catchup processing occurs when a client is connected to a read stream file after the stream file has started reading data for another client.

The client added to a running read stream may require blocks which are physically prior to the block currently being read. In this case there are two possibilities:

1. The client may be forced to wait until all currently running clients have finished reading that volume of the stream file.
2. The volume may be rewound to allow the new client to `catch up'. This causes any other clients to be delayed until the new client has read blocks up to the original read position.

Which of these alternatives to choose is specified by the CATCHup parameter. If the new client happens to require blocks beyond the current read position, no special action is required, since the stream file will continue reading in the normal sequence to satisfy the current clients.

The CATCHup parameter may specify one of the following catchup policies:

Always
Always allows a new client to catch up.

Never
Never allow a new client to catch up. The client will wait until all other currently reading clients have completed.

Within
allows the client to catchup depending on the following parameter settings:

Blocks
number of blocks specifies that the client will catchup if the first block of their file is behind the current stream file position by *number of blocks* or less.

Duration
time period specifies that the client will catchup if it is added to the stream task within this *time period*. *time period* is specified as a time interval as described by "Specifying a Time Interval or Period" on page 161.

RETainPerioD
retain period specifies the retention period of stream files for the *stream name*. *retain period* is specified as a number of days up to 9999. This parameter value is not used by MVS allocation processing when allocating the stream file. It is used to indicate to ExHPDM how long it should retain the stream file before expiry it and making it available for deletion as performed by ADMIN SCANEXPIRE processing.

Note: When a stream file *retain period* is reached or exceeded then the stream file is only expired if all clients in the stream file are also expired.

WAIT
WAIT specifies the action to be taken if a client connection cannot be made immediately to a stream task or a new stream task cannot be started. If there are any other streams available to the client as specified on the CLASS STREAM parameter then these will be attempted prior to proceeding with the WAIT action.

YES
specifies that connections that are requesting the use of the STREAM keyword should wait if they cannot be serviced by an existing stream task and the limit specified by the CONCURRENT parameter has been reached or the stream tasks fails to allocate a device.

RETRYperiod
retry period specifies a time period that the restarting of the stream tasks should be reattempted when it fails to allocate the necessary resources for the stream file. *retry period* is specified as a time interval as described by "Specifying a Time Interval or Period" on page 161. If a RETRYperiod is not specified then the stream task will

only be attempted for restart when the next client connection enters the ExHPDM server for the same stream or an existing connection for the same stream completes.

The minimum value that may be specified for retry period is 60seconds. The maximum value is 1day (or 86400seconds).

NO

If WAIT(NO) is specified then all connections made to the stream will fail if the stream task cannot be started.

Note: The use of the STREAM WAIT(YES) parameter is recommended over using the DEVICE ALLOC(WAIT) parameter. The use of DEVICE ALLOC(WAIT) can cause ENQs on SYSZTIOT that can hold up allocation of other stream tasks in the ExHPDM address space. Refer to“DEVICE Keyword” on page 107 for details. WAIT(YES) is the default.

SEParateby

specifies that ExHPDM separates stream task jobs by the following subparameters.

JOBname

causes connections in this stream to be separated according to the jobname of the client job from which they are made.

USER

causes connections in this stream to be separated according to the user name of the job from which they are made.

DSNLEVEL

causes connections in this stream to be separated according to the specified data set name qualifier level of the connecting client data set name. A range of levels may be supplied to allow portions of the client data set name to be selected.

Positive values may be specified, indicating that the qualifier is selection from the High Level Qualifier (HLQ) to the Lower Level Qualifier (LLQ). Negative values may be specified indicating that the qualifier is selection from the LLQ to the HLQ. DSNLEVEL(1) is the HLQ, DSNLEVEL(-1) is the LLQ.

Values may be selected from -21 to 21. DSNLEVEL(-21) is the same as specifying DSNLEVEL(1), DSNLEVEL(21) is the same as specifying DSNLEVEL(-1). If a qualifier is indicated that does not exist then the closest qualifier to the value is selected. For example, for the client data set’OWPL.TEST.DATASET’ a qualifier specification of DSNLEVEL(21) will perform selection on the qualifier DATASET. This qualifier will be selected for all values from DSNLEVEL(3) to DSNLEVEL(21).

Also,’0’ is allowed as the second part of the range, but not as the first part. For example, DSNLEVEL(5:0) is equivalent to DSNLEVEL(5). DSNLEVEL(0) is not a valid qualifier.

In the range format, the two parts of the range may be in any order. That is, DSNLEVEL(1:8) is equivalent to DSNLEVEL(8:1). Ranges with mixed

positive and negative values are not allowed. For example, DSNLEVEL(-1:1) is an error.

In the list format, all specified qualifiers are used in the matching process, and any order is acceptable. For example, valid specifications could be DSNLEVEL(-1 2 3) and DSNLEVEL(-1:-2 1:2 -5:-6).

For the client data set ’OWPL.QUAL2.QUAL3.QUAL4’ the following qualifiers will be used for the indicated DSNLEVEL:

- DSNLEVEL(1) uses OWPL
- DSNLEVEL(-1) uses QUAL4
- DSNLEVEL(1:3) uses OWPL, QUAL2 and QUAL3
- DSNLEVEL(-1) uses QUAL4
- DSNLEVEL(1 -1) uses OWPL and QUAL4
- DSNLEVEL(1:2 -1:-2) uses OWPL, QUAL2 and QUAL3, QUAL4.

STREAM Keyword Examples

Example 1

The following example defines a stream named TLDSSTRM, whose stream file name is made up using a mixture of fixed strings and symbolic substitution.

You should also consider the use of '&&' in fields that are to be dynamic names for the allocation of the new stream file. The stream file will be read or written to the device list specified by the device definition T1DEV. A maximum of 10 active connections are permitted to this stream definition by stream class. When this limit is reached, the connection waits **(WAIT(YES RETRY(60)))**. The stream is attempted for retry at 60 second intervals if, for example, a device cannot be obtained by the stream task.

When the connection limit is exceeded, another stream task (read or write) is started to the maximum value specified in the **CONCURRENT READ** or **CONCURRENT WRITE** parameters. All connection activity is to be logged in the ExHPDM server log.

```
STREAM(TLDSSTRM
DSN(DB2PROD.&SYSNAME..&JOBNAME..T&&LHHMMSS..BKUP)
DEVICE(T1DEV) CONNECTIONS(10)
CONCURRENT(READ(5) WRITE(5)) WAIT(YES RETP(60)) LOG(YES) )
```

Example 2

In the following example, the stream definition for stream TLSTRM, permits 50 connections before a new stream task is started. Only one stream read task is permitted while up to 5 write tasks are permitted. If a resource is not available when a client connects, it is failed through the use of the **WAIT(NO)** parameter. The stream will use the default for the stream device specification.

```
STREAM(TLSTRM
DSN(OPERDEPT.&SYSNAME..&JOBNAME..HPDMETL.STREAM)
WAIT(NO) CONNECTIONS(50)
CONCURRENT(READ(1) WRITE(5) ))
```

Example 3

In the following example, the stream DBASTRM permits 43 connections before a new stream task is started. If a resource is not available when a client connects, it is failed through the use of the **WAIT(NO)** parameter. The stream will use the default for the stream device specification.

```
STREAM(DBASTRM DSN(IMSDEPT.D&&WDAY..NIGHT.BCKUP)
WAIT(NO) CONNECTIONS(43) )
```

Example 4

In the following example, the stream definition specified uses a device definition named RWDEV.

```
STREAM(REDST3A DSN(IMSPROD.CADENG.DAILY.BACKUPS)
DEVICE(RWDEV) )
```

Example 5

In the following example, the stream definition is specified as not cataloging the stream file name in the MVS catalog. The use of **CATALOG(NO)** should only be specified in special circumstances. It is not recommended unless ExHPDM is being used as a External Data Manager with CA–1.

```
STREAM(REDST3B CATALOG(NO)
DSN(APPLIED.PHYSICS.RDDEPT.WEEKLY.BKUP) DEVICE(REDDEV))
```

Example 6

In the following example, stream definition REDST3C permits up to 80 connections and may use load balancing through the use of the stream class EFI and the stream device REDMEDIA definition (if available). EFI is not required if the **CONNECTIONS** parameter is specified. The **CONNECTIONS** parameter allows load balancing on its own. For a new stream file (write), the tape media is expired in 10 days.

```
STREAM(REDST3C DSN(SEISMIC.CAP.D&&DAY&&MON..Y&&YR4..BACKUP)
DEVICE(REDMEDIA) CONNECTIONS(80)
RETAINPERIOD(10))
```

Example 7

In the following example, stream definition REDSTR2 permits up to 10 connections and may use load balancing through the use of the stream class EFI and the stream device REDMEDIA definition (if available). EFI is not required if the CONNECTIONS parameter is specified. The CONNECTIONS parameter allows load balancing on its own. For a new stream file (write), the tape media is expired in 200 days.

When performing read processing the stream task will prompt the operator with msg SOV06500I/SOV06501I before reading the first client block. If 10 connections are made before the operator replies to the message the stream task will commence reading. If a client connection starts after the stream task has begun reading then force the stream task to reposition to the clients starting block if it has already been passed.

```
STREAM(REDSTR2 DSN(SYSTEM.BACKUP)
DEVICE(REDMEDIA) CONNECTIONS(10)
RETAINPERIOD(200)
READSYNCH(
STARTWHEN(OPERREPLY TOTALCONNECTIONS(10))
CATCHUP(ALWAYS)))
```

Example 8

This example demonstrates the effect of the SEParateby(USER) parameter

Consider the following three jobs submitted at roughly the same time. Jobs “DMPPRD1” and “DMPPRD2” are production DFSMSdss dump jobs submitted by user “PRDUSR” and job “JUSER1” was submitted by an applications programmer to perform a DFSMSdss logical backup of his own data sets.

```
//DMPPRD1 JOB  jobcard info
//* Production full volume backup job
//STEP001      EXEC PGM=SOVDSSU,REGION=4M
//SYSPRINT     DD SYSOUT=*
//DD01         DD UNIT=SYSDA,VOL=SER=HPDM01,DISP=OLD
//DD02         DD UNIT=SYSDA,VOL=SER=HPDM02,DISP=OLD
//OUT1         DD SUBSYS=SOV,DSN=DUMP.PROD.FULL.HPDM01,
//            DISP=(,CATLG,DELETE)
//OUT2         DD SUBSYS=SOV,DSN=DUMP.PROD.FULL.HPDM02,
//            DISP=(,CATLG,DELETE)
//SYSIN        DD *
             PARALLEL
             DUMP FULL INDD(DD01) OUTDD(OUT01)
             DUMP FULL INDD(DD02) OUTDD(OUT02)

//DMPPRD2 JOB  jobcard info
//* Application full volume backup job
//STEP001      EXEC PGM=SOVDSSU,REGION=4M
//SYSPRINT     DD SYSOUT=*
//DD01         DD UNIT=SYSDA,VOL=SER=HPDM03,DISP=OLD
//DD02         DD UNIT=SYSDA,VOL=SER=HPDM04,DISP=OLD
//OUT1         DD SUBSYS=SREM,DSN=DUMP.APP.FULL.HPDM03,
//            DISP=(,CATLG,DELETE)
//OUT2         DD SUBSYS=SREM,DSN=DUMP.APP.FULL.HPDM04,
//            DISP=(,CATLG,DELETE)
//SYSIN        DD *
             PARALLEL
             DUMP FULL INDD(DD01) OUTDD(OUT01)
             DUMP FULL INDD(DD02) OUTDD(OUT02)

//JUSER1 JOB jobcard info
//* backup my current development datasets
//BACKUP EXEC PGM=SOVDSSU,REGION=4M
//SYSPRINT     DD SYSOUT=*
//OUTDD        DD DISP=(,CATLG,DELETE),SUBSYS=SREM,
//            DSN=JUSER.APP.DUMP
//SYSIN        DD *
             DUMP DATASET(INCLUDE(JUSER.APPL1.**)) -
             OUTDDN(OUTDD) TOLERATE(ENQFAILURE)
```

Consider the ExHPDM parameter file containing the following configuration:

```
SELECT(SELECT1 DSN(**) CLASS(CLASS1))
CLASS(CLASS1 STREAM(STRM1))
STREAM(STRM1 SEP(USER))
```

Based on this SEParateby parameter, the effect would be to cause jobs “DMPPRD1” and “DMPPRD2” to be written to the same stream file and managed by the same stream task. A separate stream task would manage the third job submitted by the applications programmer. This is because jobs “DMPPRD1” and “DMPRPRD2” are submitted by the same userid and the third job “JUSER1” is submitted by a different userid.

Consider the ExHPDM parameter file containing the following configuration:

```
SELECT(SELECT1 DSN(**) CLASS(CLASS1))
CLASS(CLASS1 STREAM(STRM1))
STREAM(STRM1 SEP(DSNLEVEL(2)))
```

Based on this SEParateby parameter, the effect would be to cause the client data sets for the dumps to DUMP.PROD.FULL.HPDM01 and DUMP.PROD.FULL.HPDM02 to be written to the same stream file. DUMP.APP.FULL.HPDM03, DUMP.APP.FULL.HPDM04 and JUSER.APP.DUMP would be written to the another stream file since the second level qualifier is the same for each of these.

Consider the ExHPDM parameter file containing the following configuration:

```
SELECT(SELECT1 DSN(**) CLASS(CLASS1))
CLASS(CLASS1 STREAM(STRM1))
STREAM(STRM1 SEP(DSNLEVEL(-1)))
```

Based on this SEParateby parameter, the effect would be to cause all of the client data sets for the dumps to go to different stream files. This is because none of the low level qualifiers are the same.

TMS Keyword

The TMS keyword is a nonstream related optional keyword that permits the identification of the installation tape management system. The TMS keyword identifies a loadable routine that is invoked by the ExHPDM server for each tape volume that is made available from a deleted stream file. Refer to “Using ExHPDM with Your Tape Management System” on page 48 for more information.

Syntax

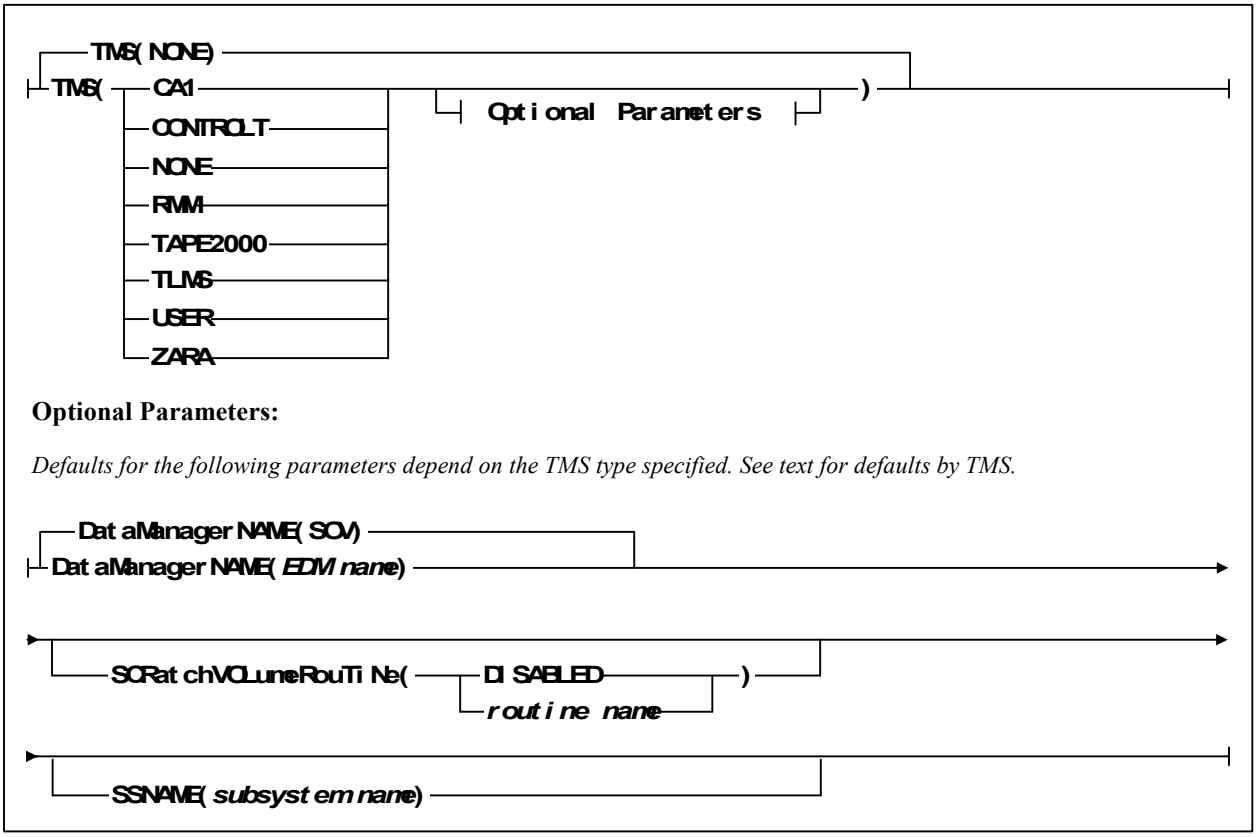


Figure 26. TMS Keyword

Keyword Name

TMS
initiates the TMS keyword.

Parameters

Tape management system names are as follows:

NONE
specifies that no External Data Manager processing is done.

NONE is the default.

CA1
specifies CA–1 as the TMS. In this case, the default scratch volume routine is TMSTMSTV.

TLMS
specifies CA–TLMS as the TMS. In this case, the default scratch volume routine is TLMSSEDM.

ZARA
specifies ASG-Zara, formerly known as AutoMedia, as the TMS. The default scratch volume routine is SOVAMTEX and is supplied as part of the ASG-Zara installation. The default TMS subsystem name is ASG-Zara.

CONTROLT
specifies Control–T as the TMS. The scratch volume routine is SOVCTTEX and is supplied as part of the Control–T installation.

RMM
specifies DFSMSrmm as the TMS. The default scratch volume routine name is EDGDFHSM.

TAPE2000
specifies Tape2000 as the TMS. The default scratch volume routine name is SOVT2TEX and is supplied as part of the TAPE2000 installation. The default TMS subsystem name is TLFS.

USER
specifies to use the loadable routine specified by the SCRatchVOLumeRouTiNe parameter.

SCRatchVOLumeRouTiNe
specifies the 1 to 8 character name of a reusable load module available in the system library search sequence that is loaded for each invocation of a stream file delete or scratch handling situation. **DISABLED** indicates no SCRatchVOLumeRouTiNe will be used.

The scratch routines for ASG-Zara, Control–T, and Tape2000 are supplied by the vendors of these products specifically for ExHPDM. The scratch routines for CA–1, TLMS, and RMM are supplied by the vendors of these products as a general interface. See each product's installation instructions for a full description.

If *routine_name* is set to DISABLED, then no routine is called. Otherwise, the default routine name depends on the specified TMS, as follows:.

Table 5 ***Default TMS SCRatchVOLumeRouTiNe***

TMS	SCRatchVOLumeRouTiNe
CA1	TMSTMSTV
TLMS	TLMSSEDM
ZARA	SOVAMTEX
CONTROLT	SOVCTTEX
RMM	EDGDFHSM
TAPE2000	SOVT2TEX
USER	DISABLED
NONE	DISABLED is the only allowed value

When specified the routine specified must obey the following conventions:

Parameters are passed by general purpose register 1 and are in SP 130 RMODE 24 in the execution key of ExHPDM.

GPR1 points to the following parameter list:

+0 @SCRPARMS
+4 @RC

@RC is the address of a full word that can be set by the scratch routine to indicate the outcome of the scratch processing. The top bit of this address is set. Any nonzero return value will result in message SOV06957 being displayed. If the return code in this field is zero on return from the scratch routine then the value returned in general purpose register 15 will be used.

SCRPARMS is a structure that contains the scratch information as shown in the following table:

Table 6. SCRatchVOLumeRouTiNe calling parameters.

Offset	Type/length	Description
+X'00'	CL6	Blank padded scratch volser.
+X'06'	XL2	2 Bytes of Flag values: Byte 1: X’80’ volume is being scratched (purged). Byte 2: Currently unused.

Table 6. SCRatchVOLumeRouTiNe calling parameters.

Offset	Type/length	Description
+X'08'	XL1	Parameter list version: X'00': first version of parm list. X'02': second version of parm list.
+X'09'	CL4	Blank padded TMS subsystem name as specified by the SSNAME keyword in the ExHPDM parameter file. Only supplied if parameter list version is X'02' and above.
+X'0D'	CL44	First data set on volser that is being scratched (blank padded). Only supplied if parameter list version is X'02' and above.
+X'39'	CL8	Blank padded Data Manager name as specified by the TMS DataManagerNAME keyword in the ExHPDM parameter file. Only supplied if parameter list version is X'02' and above.

When a program error is encountered during processing in the routine called by the **SCRatchVOLumeRouTiNe** parameter, ExHPDM issues an error message to the MCS console indicating the routine has failed and disables the routine. No further calls are made to this routine until ExHPDM is restarted or a SET **PRM** operator command issued.

SSNAME

subsystem name specifies the 1 to 4 character name of the TMS subsystem to be passed as an extra parameter to the scratch volume routine. The default value for ASG-Zara is ZARA. The default for Tape2000 is TLFS. This value is ignored for NONE. Message 3037 will be displayed if it has been supplied for NONE:

SOV03037W TMS *subsystem_name keyword_value* ignored for TMS NONE.

There is no default for other Tape Management Systems. If a subsystem name is supplied or defaulted, then a test is made during parameter processing (ExHPDM startup or SET PRM) to see if the value is defined as an MVS subsystem. If no subsystem is located for the name, then the warning message 3036 is displayed:

SOV03036W TMS SSNAME *subsystem_name* is not defined as an MVS subsystem.

This does not prevent the subsystem name from being used or passed to the SCRVOLRTN, as the subsystem might be added dynamically.

DataManagerNAME

EDM name is the 1 to 8 character name that ExHPDM has been identified to the TMS as a data manager (i.e., the EDM name of ExHPDM as defined to the TMS). If no name is used to define ExHPDM, then leave this value as the default. This value is ignored for NONE. Message 3037 will be displayed if it has been supplied for NONE.

Note: TAPE2000 is the only current user of this value.

DataManagerNAME(SOV) is the default.

TMS Keyword Examples

Example 1

In the following example, ExHPDM is identified to the CA–1 tape management system as an External Data Manager. The CA–1 routine TMSTMSTV is loaded and called from the system libraries by ExHPDM when stream file tape volumes are to be made available for scratching. Note that the standard External Data Manager scratch interface for CA–1 is TMSTMSTV.

```
TMS(CA1)
```

Example 2

In the following example, ExHPDM is identified to the CA–1 tape management system as an External Data Manager. All tape scratch processing is formed by an installation tape scratching routine called CA1EXTRA. The routine CA1EXTRA is loaded and called from the system libraries by ExHPDM when stream file tape volumes are to be made available for scratching.

```
TMS(CA1 SCRVOLRTN(CA1EXTRA))
```

Example 3

In the following example, all tape scratch processing is formed by an installation tape scratching routine called USERSCR. ExHPDM loads and calls a loadable routine called USERSCR from the system libraries when stream file tape volumes are to be made available for scratching.

```
TMS(USER SCRVOLRTN(USERSCR))
```

Example 4

In the following example, No specialized external scratch tape processing will be used.

```
TMS(NONE)
```

Example 5

In the following example, all tape scratch processing will be formed by CA–1 tape management system but the installation tape scratching routine is disabled. ExHPDM will not attempt to call the routine specified by the SCRatchVOLumeRouTiNe parameter.

```
TMS(CA1 SCRVOLRTN(DISABLED))
```

Example 6

In the following example, ExHPDM is identified to the CONTROL–T tape management system as an External Data Manager. The CONTROL–T routine SOVCTTEX is loaded and called from the system libraries by ExHPDM when stream file tape volumes are to be made available for scratching.

```
TMS(CONTROLT)
```

Example 7

In the following example, ExHPDM is identified to the ASG-Zara (formerly AutoMedia) tape management system as an External Data Manager. The ASG-Zara routine SOVAMTEX is loaded and called from the system libraries by ExHPDM when stream file tape volumes are to be made available for scratching.

```
TMS(ZARA)
```

Specifying a Time Interval or Period

Time intervals, also known as time periods, for various ExHPDM keyword parameters may be specified in the format described in the following syntax diagram.

Syntax

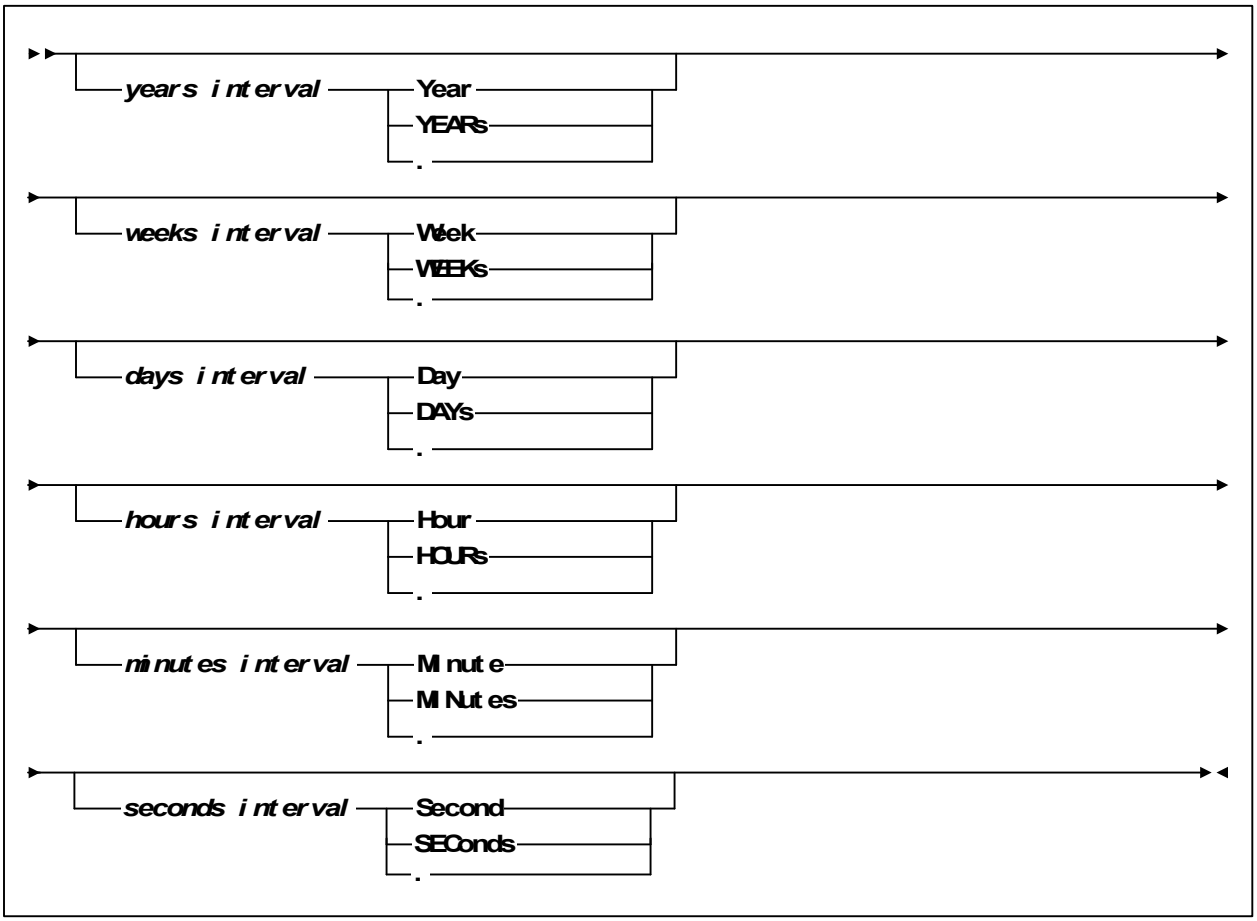


Figure 27. Time Interval

Keyword Name

There is no particular keyword associated with a time interval. Various ExHPDM keywords reference this time interval definition. In general time intervals and periods are referenced in the syntax diagrams with names suffixed by *interval* or *period*. Some ExHPDM keywords limit the size of time interval specified. This information is associated with and therefore documented with the keyword.

Parameters

Time interval units may be specified as follows:

Years
specifies the number of years to sum for the time interval. A year is defined as 365.25 days for the purpose of computing the time interval.

Weeks
specifies the number of weeks to sum for the time interval.

Days
specifies the number of days to sum for the time interval.

Hours
specifies the number of hours to sum for the time interval.

Minutes
specifies the number of minutes to sum for the time interval.

Seconds
specifies the number of seconds to sum for the time interval.

Plurals are allowed for longer forms of units to make the value more readable. For example, SECONDS or SECS but not SS. Correct usage of grammar is not verified when using plurals. For example 1minutes is allowed.

Each timer interval unit may be separated by a '.' in addition to the unit value. For example 5minutes.1second is equivalent to 5minutes1second. Where a time interval unit is not specified then a '.' is required. For example 5.1.

Each of the time interval units as specified above is optional. Where a time unit is not explicitly specified its unit is implied from the previous or next unit. Where no units are specified at all then they start from the lowest unit on the right working to the left with successively larger units:

nn
Number of seconds

nn.nn
Minutes then seconds. This pattern continues up to *nn.nn.nn.nn.nn.nn* which is years, weeks, days, hours, minutes then seconds.

nn.nnU
U indicates any of the specifiable units (Seconds, Minutes, Hours, Days, Weeks, Years). The leading digits without explicit units are assigned successively larger units from right to left. For example 1.2.3h is 1 week, 2 days, and 3 hours.

nnU.nn
U indicates any of the specifiable units (Seconds, Minutes, Hours, Days, Weeks, Years). Digits without explicit units which follow digits with units are assigned successively smaller units from left to right. For example 2h.30.600 is 2 hours, 30 minutes, and 600 seconds.

Combinations of these rules are also valid, with the *nnU.nn* rule having precedence over the *nn.nnU* rule. For example 2.1d.3.4s is 2 weeks, 1 day, 3 hours (not 3 minutes) and 4 seconds.

Examples

Example 1

To specify a time interval of 10 seconds:

10s

Example 2

To specify a time interval of 1 minute:

1minute

Example 3

To specify a time interval of a million seconds:

1000000secs

Example 4

To specify a time interval of 1 year, 20 weeks, 5 days, 2hours:

2h20w5d1y

Example 5

As per example 4 using non explicit intervals:

1.20.5.2h

Example 6

To specify a time interval of 20 days, 10 hours, 9 minutes. In this example the lone digit '9' is assigned a unit less than the *lowest* explicit unit. The '.' separating the units is optional:

10hours.20days.9

Example 7

The following is an error since the '5' follows an explicit seconds unit. A unit less than a second is not allowed:

2s.5.10h

Example 8

The following is an error since the ‘5’ precedes the maximum unit value of a year:

```
5.20y
```

Example 9

The following is an error since a unit can only be specified once:

```
10Hours20Days9Hours
```

File Example

The following is an example of a startup parameter file. This parameter file becomes a SOVPRM00 member.

```
/*================================================================
Member:SOVPRM00
MV System Name:&sysname
MVS Subsystem Name:&sovssn
PRODUCT ID:&sovpid
FMID:&sovfmid
Purpose:
Describes the ExHPDM server address space's resources,
selection rules, stream , class, and device definitions.

      This member must be tailored for your needs.

Notes:
     - parameter data is not limited to 72 characters.
     - lowercase is permitted for all keywords and parameters.
     - column numbering must NOT be used.

================================================================*/
/*----------------------------------------------------------------
          Database and Journal Definitions
----------------------------------------------------------------*/
```

database(exhpdm.database)

```
/*
MANDATORY specification.
Specifies the VSAM KSDS data set to be used as the database for the ExHPDM servers. The database
is defined in installation job DEFDBASE. If the database is to be shared among multiple ExHPDM
servers, it must be defined to VSAM withSHAREOPTIONS(4,4).
*/
```

backupdsn(exhpdm.database.backup unit(sysallda))

```
/*
(optional specification) specifies a MVS data set name prefix to be used to form the backup data set
for this database. The prefix must conform to MVS naming standards and is limited to 35
characters.ExHPDM will provide a unique suffix to this string to form the name of the backup data
set.This data set is used for the ADMIN DATABASE BACKUP command. Although optional, it is
```

recommended that a database backup data set is used. The UNIT(SYSALLDA) determines where the file is to be allocated. Only DASD specifications should be used.
***/**

journal(exhpdm.jrnl.&sovssn..&sysname)

/*
(optional specification) Specifies the name of the database journal data set. This is a fully qualified data set name. The journal data sets cannot be shared among ExHPDM servers. The journal data set must be allocated using the DEFDBASE installation job.

For example, if there are three ExHPDM servers sharing the same database, three journal data sets must be defined. This definition uses symbolic substitution to derive a unique name based on the ExHPDM server name and the current MVS system name. The result MIGHT be a journal name of 'EXHPDM.JRNL.SREM.MVSO'. Although optional it is recommended that a journal data set is used.
***/**

backupjournal(exhpdm.jrnl.bkup.&sovssn..&sysname unit(sysallda))

/*
(optional specification) specifies a MVS data set name prefix to be used to form the backup data set for this journal. The prefix must conform to MVS naming standards and is limited to 35 characters. ExHPDM will provide a unique suffix to this string to form the name of the backup data set. This data set is used for the ADMIN DATABASE BACKUP command. Although optional, it is recommended that a database backup journal data set is used. This definition uses symbolic substitution to derive a unique name based on the ExHPDM server name (&sovssn) and the current MVS system name (&sysname). The result MIGHT be a backup journal data set name of
'EXHPDM.JRNL.BKUP.SREM.MVSO'.
***/**

```
LKEYINFO(CUSTOMER(custname)
         SITE(siteno)
         EXPIRY(expdate)
         PRODUCT(prodname)
          KEY(keyvalue)
```

monitor(database(threshold(70) warnevery(percentageincrement(5)))
journal(threshold(85) warnevery(interval(2MIN))))

/*
(optional) Specifies that warning message SOV09218W is to be
issued when the database is more than 70% full. The message will be reissued every 5% increase there after. Specifies that warning message SOV09219W is to be issued when the journal is more than 85% full. The message will be reissued every two minutes there after.
***/**

/*----------
Logfile, SAF, Request Limits and TMS definitions
----------*/

logfile(sysout(e))

/*
(optional) Specifies that the ExHPDM server log is to be written to a JES data set (SYSOUT) to class

'E'. Specification to SYSOUT is the preferred way to manage the ExHPDM server log because it can be viewed with any JES spool browser.
***/**

prefix(message(&sysclone&sovssn(-1:1))
command(hpdm&sovssn))

/*
(optional)
Specifies the character strings to be used for ExHPDM message prefixes and the ExHPDM operator command string.

The default message prefix is 'SOV'. In this example,symbolic substitution has been used to form a message prefix from message(&sysclone&sovssn(-1:1)). &sysclone and the last character of the ExHPDM server's MVS subsystem name (&sovssn(-1:1)). For example id &sysclone is 'VO' and &sovssn is 'SREM' then the message prefix will be 'VOM'.

'Command(hpdm&sovssn)' defines the ExHPDM operator command string. This string is limited to 15 characters. The default command prefix is the MVS subsystem name of the ExHPDM server. In this example, symbolic substitution has been used to form an ExHPDM command prefix of 'hpdmsrem' where &sovssn is 'SREM'.

Note that blanks may be enclosed in quotes for this command string. However care MUST be exercised to avoid duplication of the names of valid command or other MVS components and operating system software.
***/**

saf(class($exhpdm))

/*
(optional)
Specifies a user defines MVS SAF resource class to be used for ExHPDM stream class and stream name validation. The default is MVS's 'FACILITY' class. Note that definition of a specialised SAF class is a matter of security integrity.

In this example, "saf(class($EXHPDM))", the $EXHPDM class is to be used for resource validation.
***/**

tms(ca1 scratchvolumeroutine(tmstmstv))

/*
(optional)
Specifies the acknowledgement to a tape management system (a TMS) that ExHPDM will perform external data management of its tape volumes. The name of the tape management system is specified. Additionally, the name of the system available routine to perform
tape volume scratching is also specified.

In this example, "tms(ca1) scrvolrtn(TMSTMSTV))', ExHPDM is defined to tape management system called 'CA1'. The scratch routine to be used is called "TMSTMSTV". Note that this is the routine for CA-1 TMS.
***/**

/*----------
Setting REQUEST limits.
Note that REQUEST limits are NOT dynamically refreshable using the 'SET PRM=nn' command. The ExHPDM server MUST be restarted to use modified values.
----------*/

request(admin maxlimit(10) log(yes))

```
/*
(optional)
Specifies that a maximum of 10 SOVADMN requests can be active simultaneously. The system print
output is also to be directed to the ExHPDM server log.
*/
```

request(connect maxlimit(200) log(yes))

```
/*
(optional)
Specifies that a maximum of 200 client connections are to be permitted. The message output is also
to be directed to the ExHPDM server log.
*/
```

request(opercmd maxlimit(5) log(yes))

```
/*
(optional)
Specifies that a maximum of 5 ExHPDM operator commands can be active simultaneously. The
console output is also to be directed to the ExHPDM server log.
*/

/*----------------------------------------------------------------------
Select Rules (optional)

Select rules are used to route connections to a specified stream class.
----------------------------------------------------------------------*/
```

select(tlrule class(operprod) datasetname(maynard.tl.full.*))

```
/*
route connection to class 'operprod' when client data set name matches 'maynard.tl.full.*'.
*/
```

**select(tlrule1 class(lanback)
datasetname(maynard.tl.rule1.*.sec1))**

```
/*
route connection to class 'lanback' when client data set name matches 'maynard.tl.rule1.*.sec1'.
*/
```

```
select(tlrule2 class(db2prod)
datasetname(maynard.tl.rule2.*.sec2) )

  /*
  route connection to class 'db2prod' when client data set name matches 'maynard.tl.rule2.*.sec2'.
  */


select(tldba class(dbaprod) job(dbaprod*))

 /*
 route connection to class 'dbaprod' when client job name matches 'dbaprod*'.
 */


select(redru3a class(imsprod)
datasetname(maynard.rd3a.full.*) )

  /*
  route connection to class 'imsprod' when client data set name matches 'maynard.rd3a.full.*'.
  */


select(redru3b class(redcl3b)
datasetname(maynard.rd3b.full.* ) )

  /*
  route connection to class 'redcl3b' when client data set name matches 'maynard.rd3b.full.*'.
  */


select(redru3c class(seismic)
datasetname(maynard.rd3c.full.*) )

   /*
   route connection to class 'seismic' when client data set name matches 'maynard.rd3c.full.*'.
   */


select(red3job class(imsprod) job(rq74*) )

  /*
  route connection to class 'imsprod' when client jobname matches 'rq74*'.
  */
```

select(red3step class(imsprod) step(dbap*))

```
/*
route connection to class 'imsprod' when client jobstep matches 'dbap*'.
*/
```

select(red3proc class(imsprod) pstep(cga*))

```
/*
route connection to class 'imsprod' when client procstep matches 'cga*'.
*/
```

**select(seismic class(imsprod) user(maynard) sys(ecc1o)
dsn(seismic.weekly.backup.*))**

```
/*
route any connection with an owner of 'maynard' with a sysname of 'ecc10' and a data set name
that matches 'seismic.weekly.backup.*' to a stream class called 'imsprod'.
*/
```

select(prodsatt exclude jobname(satt*))

```
/*
don't permit any connections to this ExHPDM server whose jobname begins with 'satt' unless they
match one of the previous SELECT statements.
*/

/*--------------------------------------------------------------------
            Stream Class Definitions.
 --------------------------------------------------------------------*/
```

class(operprod stream(tlstrm))

```
/*
stream class for operations MVS operating system backup work.
*/
```

class(db2prod stream(tldsstrm) expectedflowincrease(3M))

```
/*
stream class for all db2 backup work. Each connection is assumed to be capable of delivering 3 MB
per second from the disk arrays.
*/
```

```
class(lanback expectedflowincrease(0) stream(tlstrm) )

   /*
   stream class for LAN backup work. There is no load balancing used.
   */


class(dbaprod expectedflowincrease(0) stream(dbastrm) )

   /*
   stream class for dba backup work. There is no load balancing used.
   */


class(imsprod stream(redst3a) )

   *
   Stream class used for all IMS production backups These applications will be backed up using
   Redwood type 'A' cartridges (10 MB).
   */


class(redcl3b stream(redst3b) expectedflowincrease(1500K ))

   /*
   stream class used for clients requiring medium amounts of high-capacity storage. Clients will be
   assumed to be capable of delivering 1500 KB second from the disk arrays.
   */


class(seismic expectedflowincrease(7M ) stream(redst3c) )

/*
stream class used for seismic data archiving that requires high performance and high-capacity Redwood type
'C' cartridges (50GB). It is expected that the clients are capable of delivering 7 MB second from a server's disk
array.
*/

/*-----------------------------------------------------------------
        stream definitions.
 -----------------------------------------------------------------*/


stream(tldsstrm
     dsn(db2prod.foosys.topjob.t235959.bkup)
```

```
    device(tldev) connections(10)
    concurrent(read(5) write(5)) wait(yes) log(yes) )

  /*
  defines a stream called 'tldsstrm' whose stream file name will begin with
  'db2prod.foosys.topjob.t235959.bkup'.

  The stream file will be read or written to the device list specified by the device definition 'tldev'.

  A maximum of 10 active connections is permitted to this stream definition by stream class.

  When this limit is reached, the connection will wait (wait(yes).

  When the connections limit is exceeded, another stream task (read or write) is started to the
  maximum value specified in the 'concurrent(read) or concurrent(write).

  All connection activity is to be logged in the ExHPDM server log.
  */



stream(tlstrm
    dsn(operdept.&sysname..&jobname..vtapetl.stream) wait(no)
    connections(50) concurrent(read(1) write(3) ))

    /*
    Stream definition for stream 'tlstrm'. 50 connections are permitted before a new stream task is
    started. Only one stream read task is permitted while up to 3 write task are permitted.

    If a resource is NOT available when a client connects, the connection is failed through the use of
    'wait(no)'.

    The stream will default the stream device definition.
    */


stream(dbastrm
    dsn(imsdept.d&&wday..night.backup) wait(no)
    connections(43) )

     /*
     Stream definition for stream 'dbastrm. 43 connections are permitted before a new stream task is
     started.

     If a resource is NOT available when a client connects the connection is failed through the use of
     'wait(no)'.

     The stream will default the stream device definition.
     */
```

```
stream(redst3a dsn(imsprod.cadeng.daily.backups) device(rwdev) )

   /*
   Stream definition is specified to use a device definition called 'rwdev'.
   */


stream(redst3b catalog(no)
     dsn(applied.physics.rddept.weekly.bkup) device(reddev) )

    /*
    Stream definition is specified to NOT catalog the stream file name in the MVS catalog. The use of
    'CATALOG(NO)' should be specified in special circumstances only. It is NOT recommended.
    */


stream(redst3c
     dsn(seismic.cap.d&&day&&mon..y&&yr4..backup)
     device(reddev) connections(80)  retainperiod(10) )

  /*
  Stream definition 'REDST3C' permits up to 80 connections and MAY use load balancing via the
  stream class EFI and the stream device (REDDEV) definition (if available).

  For a new stream file ( write), the tape media will be expired in 10 days.
  */

   /*----------------------------------------------------------------
  Stream Device Definitions.
  The unit specifications MUST be modified for site specific use.
  Note that specific device numbers should not be used!
   ----------------------------------------------------------------*/


device(tldev unit(ausall) devicelimit(16) default)

   /*
   Stream device 'TLDEV' will use all tape devices specified in the esoteric name 'AUSALL'. A limit
   of 16 tape devices is permitted for all usage of this stream device definition.

   This is the default stream device specification.
   */
```

device(rwdev unit(rwall) devicelimit(2))

/*
Stream device 'rwdev' will use all tape devices specified in the esoteric name 'rwall'. A limit of 2 tape devices is permitted for all usage of this stream device definition.
*/

device(reddev unit(ausred rwall) unitcount(2) flowlimit(12m) alloc(wait) compact(no) devicelimit(8) retain(write(15m) read(10s)))

*
Stream device definition 'reddev' specifies the tape devices to be used. In this example, these are named in the unit(ausred rwall) parameter as being esoteric named devices called 'ausred' and 'rwall'. The stream file will be allocated to ausred first and when exhausted, will prompt through MVS allocation retry because 'allocwait(yes)' has been specified.

If the device cannot be satisfied, the stream file allocation will attempt allocation using the name 'rwall' and so on. Should allocation still not be satisfied, the stream task will close and reject client connections.

For all stream file allocations, use two tape devices from the unit list '(unitcount(2)'. This permits premounting of tape media.

Don't permit compaction or compression at the tape device (compact(no)) and limit tape allocations for all usage of this stream device to 16 tape drives.

The stream file will remain available of the stream file device for up to 15 minutes of inactivity for write streams and 10 seconds for read streams.

Last, the devices specified in this definition have a maximum throughput of 12 MB per second (flowlimit(12M). This will be used with any EFI specification on the class statement to cause load balancing. In such cases, a new stream task will be started based on limits expressed in the concurrent statements.
*/

Chapter 7. Starting the ExHPDM Server

Start Command

The ExHPDM server can be started by either of two ways:

1. During MVS subsystem initialization, or
2. Using the MVS START operator command.

Starting the ExHPDM server during subsystem initialization is achieved through the use of the MVS parameter library member IEFSSN*xx*. If ExHPDM is started in this manner, the installed load modules must be available at SSI initialization in the MVS linklist concatenation. See “Chapter 11. Changing MVS Parameter Files” on page 283 for additional information about the MVS parameter library files to support ExHPDM.

To explicitly start the ExHPDM server, the MVS START Operator command must be used.

Warning: ExHPDM will force a **RENEW** start automatically whenever a change of maintenance, either uplevel or downlevel, is detected.

Syntax

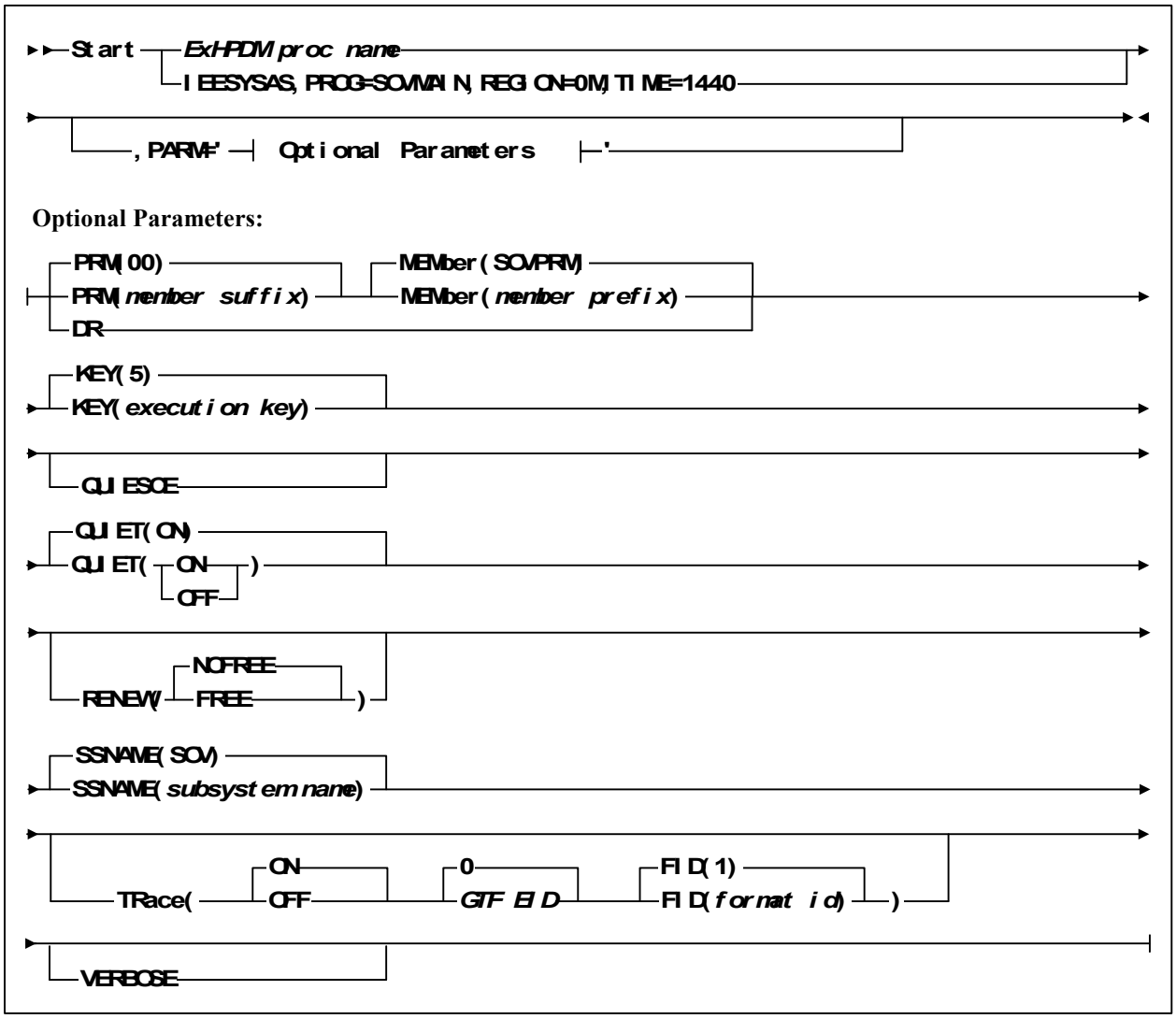


Figure 28. MVS START Command

Command Name

Start
initiates the MVS START Operator command.

Parameters

ExHPDM proc name
specifies any valid procedure as setup for the ExHPDM server. The standard MVS system address space proc IEESYSAS can be used by specifying the PROG parameter. Note that the JCL TIME and REGION must be specified when ExHPDM is started using IEESYSAS. A REGION value of 0M or 0K may be specified to allow ExHPDM to obtain as much virtual storage above and below the line as

required. A REGION value of 4M is the minimum recommended value. Refer to the *MVS JCL Reference Guide* for these keywords.

PRM

member suffix specifies the optional two character suffix for the parameter file member name. The member name is formed using the *member prefix* specified in the **MEMBER** parameter. If the STKPARMS DD is a sequential data set (non PDS), the **PRM** suffix is ignored. If the **DR** parameter is specified, **PRM** cannot be specified.

PRM(00) is the default.

MEMBER

member prefix specifies the 1 to 6 character parameter file member name prefix. The member name is formed using the suffix specified in **PRM**. If STKPARMS DD is a sequential data set (non PDS), the **MEMBER** prefix is ignored. If the **DR** parameter is specified, **MEMBER** cannot be specified.

MEMBER(SOVPRM) is the default.

DR

specifies that the ExHPDM server is to start in disaster recovery mode. In this mode there is no real ExHPDM database. The database is unavailable and usually cannot be reinstated. In disaster recovery mode, ExHPDM startup builds a dummy parameter file internally to simulate the environment for internal processing. The **SCANSTReamfile** parameter of the ExHPDM Administration Utility is used to locate and restore client data sets. If **DR** is specified, **MEMBER** and **PRM** parameters cannot be specified.

KEY

execution key specifies the PSW key for ExHPDM server in addition to the storage key used by ExHPDM when allocating common storage. Any key between 1 and 15 may be specified. However, problem keys such as KEY(8) should not be specified as this results in common area storage to be allocated in this key. The recommendation is to use the default value.

Note: If a KEY is specified that is different to the previously started ExHPDM KEY then a RENEW start is forced.

KEY(5) is the default.

QUIESCE

specifies that no access to the ExHPDM database is allowed until a SET **DATABASE RESTART** operator command is issued. This command is used when restoring a database.

QUIET

specifies whether or not additional internal diagnostic messages are to be output. This parameter should only be used when advised by StorageTek Technical support.

QUIET(ON) is the default.

RENEW

specifies that the ExHPDM MVS common area data structures including modules from the previously started ExHPDM server should be rebuilt.

ExHPDM forces a RENEW start when a new version of the product is started, a global area/execution key change is specified, or when a change of maintenance, either uplevel or downlevel, is detected.

NOFREE

specifies that the common area structures are not to be freed and are to remain allocated until the next IPL.

NOFREE is the default.

FREE

specifies that most of the common area structures are to be freed. A small amount of the common area remains allocated until the next IPL.

Warning: If the application of maintenance is held by the use of SMP/E HOLDDATA and the instructions indicate that the **RENEW(FREE)** parameter must be specified, then removal of the maintenance requires that the **RENEW(FREE)** parameter is specified when restarting ExHPDM. Removing the maintenance is either restoring the maintenance or going back to a load library where the maintenance was never applied.

SSNAME

subsystem name specifies a 1 to 4 character subsystem name defined for ExHPDM usage in IEFSSN*xx*. See “MVS Subsystem Definition (IEFSSNxx)” on page 283 for details on defining the subsystem name.

Note: When using FDRABR, use the default ExHPDM subsystem name of SOV to support the FDRABR DYNTAPE operand. If an ExHPDM subsystem name other than SOV is used, contact Innovation Data Processing Support.

TRace

turns GTF ON or OFF. This allows GTF tracing to be performed on the ExHPDM initialization processing before the operator command interface is activated. Tracing can be turned ON or OFF by the SET TRace command once the operator command interface is active.

GTF must be active for tracing TYPE=USR events. GTF tracing of ExHPDM events is automatically turned off when ExHPDM is shutdown.

trace id (GTF EID)

trace id specifies that tracing only occurs to GTF where this matches the event identifier (EID) on the USRP GTF keyword for the currently active GTF trace. This may be required to avoid tracing other nonExHPDM related GTF trace records. A value in the range x’000’ to x’3FF’ should be specified.

If USRP is not currently in effect for the TYPE=USR GTF trace, tracing always occurs. The selected *trace id* is in effect until a SET **TRACE** command is entered to change the value.

For addition details about using GTF and the USRP keyword, refer to the *MVS/ESA SP V5 Diagnosis: Tools and Service Aids.*

The default trace id is 0.

Notes: As the values specified on the GTF USRP are specified as hexadecimal values, it may be easier to use hex values for the *trace id*. A hexadecimal value is entered as x'*nnn*'. Where *nnn* matches the GTF EID.

FID

format id specifies the trace formatting appendage to use when formatting the GTF trace entries with IPCS. The *format id* may be any value from, decimal, 0 to 80. The *format id* indicates which AMDUSR*nn* is used by the IPCS GTFTRACE command. Where *nn* is the *format id* and is in the range hexadecimal 00 to hexadecimal 50 (decimal 80). The selected *format id* is in effect until a SET **TRACE** command is entered to change the value.

The default *format id* is 1.

Note: As the value to determine the AMDUSR*nn* formatting routine is specified as a hexadecimal value, it may be easier to use hex values for the *format id*. A hexadecimal value is entered as x'*nn*'.

VERBOSE

specifies that additional internal diagnostic messages are to be output to the ExHPDM LOGFILE. This parameter should only be used when advised by StorageTek Software Support. Specifying VERBOSE can result in large amounts of output to be generated to the ExHPDM LOGFILE

ExHPDM VERBOSE messages are only output when log file messages are being directed to a data set or to SYSOUT. No VERBOSE messages are output when the LOGFILE WTO option has been specified. Refer to "LOGFILE Keyword" on page 117.

START Command Examples

Example 1

The following example initiates a ExHPDM server startup using the default subsystem name of SOV, the IEESYSAS procedure, and the default startup parameter of PRM=00.

```
START IEESYSAS,PROG=SOVMAIN,TIME=1440,REGION=4M
```

Example 2

The following example initiates a ExHPDM server startup using a subsystem name of VTP2, the IEESYSAS procedure, the startup parameter file SOVPRMTS, and initiates GTF tracing.

```
S IEESYSAS,PROG=SOVMAIN,PARM='SSNAME(VTP2) PRM(TS)
TR',TIME=1440,REGION=4M
```

Example 3

The following example initiates a ExHPDM server in disaster recovery mode.

```
S EXHPDM,PARM='SSNAME(SOVW) DR'
```

Example 4

The following example initiates a ExHPDM server startup using the procedure SOVPROC, the default subsystem name of SOV, and the default startup parameter file SOVPRM00.

```
S SOVPROC
```

An example of a startup procedure for ExHPDM is shown in the following figure. This same startup JCL is located in the installation sample library STKSAMP member SOVPROC.

```
//SOVPROC  PROC
//*-------------------------------------------------------------------
//*
//*             Sample startup procedure for ExHPDM
//*             ----------------------------------
//*
//* Change 'hlq1' to the high level qualifier(s) used during the
//* install of ExHPDM.
//*
//* This sample startup proc uses the sample ExHPDM parameter member
//* SOVPRM00 installed in hlq1.STKSAMP.  An alternate parameter
//* member may be selected by changing the PRM(00) on the EXEC PARM
//* to the appropriate suffix value.  For example, to select
//* SOVPRM01 use PRM(01)
//*
//*-------------------------------------------------------------------
//*
//*
//SOVPROC EXEC PGM=SOVMAIN,PARM='PRM(00)',TIME=1440,REGION=0M
//*
//*-------------------------------------------------------------------
//*
//* StorageTek load library -
//*
//* Required if not in system linklist concatenation.
//*
//* This library must be APF authorized.
//*
//* Change hlq1.STKLOAD to the installed load library name.
//*
```

```
//*-------------------------------------------------------------------
//*
//STEPLIB DD DISP=SHR,DSN=hlq1.STKLOAD
//*
//*-------------------------------------------------------------------
//*
//* StorageTek parameter library Data Set - optional.
//*
//* The default member name for these startup parameters is
//* SOVPRM00.
//*
//* The STKPARMS DD is required if the parameters are not in
//* SYS1.PARMLIB.
//*
//* If the SOVPRM00 member cannot be located in the STKPARMS DD or
//* the STKPARMS DD has not been specified then SYS1.PARMLIB will be
//* used.
//*
//* Change hlq1.STKSAMP to the required parameter data set name.
//*
//*-------------------------------------------------------------------
//*
//STKPARMS DD DISP=SHR,DSN=hlq1.STKSAMP
//*
//*-------------------------------------------------------------------
//*
//* Standard SYSOUT and diagnostic output - optional.
//*
//*-------------------------------------------------------------------
//*
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//*
//* Dump Data Set - optional. Change <dumpds> to the
//* required dump data set name.
//*
//*SYSMDUMP DD DISP=SHR,DSN=<dumpds>
```

Figure 29. ExHPDM Startup Example Procedure

Example 5

The following example shows the message that indicates a change in the SOVSERV1 maintenance level:

SOV03???W Global module SOVSERV1 is at maintenance level %s. RENEW will be performed for maintenance level %s.

The maintenance levels displayed are the PTFs that are influencing the change. The first field indicates the current PTF level for SOVSERV1. The second indicates the PTF level which it should be.

Example 6

If after the RENEW has been performed the global module is still not at the required level then ExHPDM will terminate. The probable reason for this failure is that an incorrect SOVSERV1 had been found in the library search sequence. This indicates an installation error. Message SOV03059E allows a maintenance level to be displayed in addition to the existing version information as follows.

SOV03059E Global module SOVSERV1 is %s %s, should be %s.

Where the fields are defined as :

- Character string to indicate a version or maintenance level
- Global module version in format *version.release.modification* or the global module maintenance level
- The version or maintenance level that the global module should be.

Example 7

When ExHPDM is started with a change in SOVSERV1 that requires a RENEW start the following message is issued.

SOV03???W Global module SOVSERV1 is at maintenance level L1P019A. RENEW will be performed for maintenance level L1P019C.

Example 8

If the LOAD of SOVSERV1 does not yield the correct maintenance level then the following SOV03059E message is produced.

SOV03059E Global module SOVSERV1 is maintenance level L1P019A, should be L1P019C.

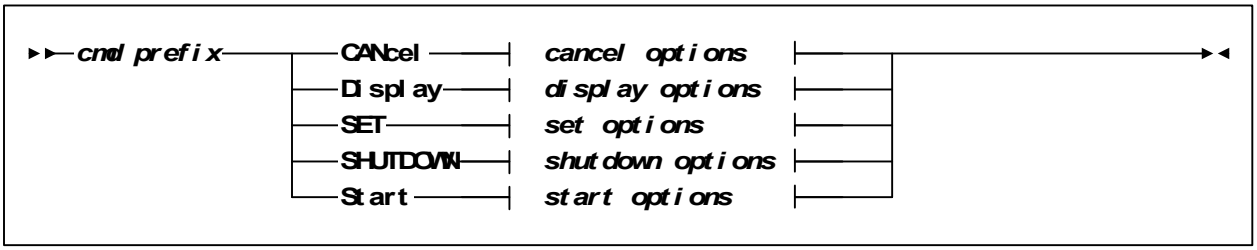
Chapter 8. ExHPDM Operator Commands

Entering ExHPDM Commands

ExHPDM operator commands control and display the status of the ExHPDM server and its clients. The ExHPDM operator commands can be entered at any MVS MCS console or other authorized MGCR(SVC34) source.

Operator commands are entered in the format as described by the following syntax diagram from the MVS MCS console:

Command Syntax



ExHPDM Operator Command Format

The *cmd prefix* is defined in the ExHPDM parameter file SOVPRM*xx*. This value is defined through the use of the PREFIX keyword of the start up parameter file. The PREFIX keyword is described on page 128.

Operator commands provide services for:

1. Canceling:
    - Stream tasks
    - Clients
    - Individual client connections
2. Displaying:
    - Stream tasks
    - Clients
    - Most recent ExHPDM LOG entries

- ExHPDM parameters in use
- ExHPDM diagnostic information

3. Setting ExHPDM session parameters
4. Shutting down ExHPDM
5. Starting a stream task.

The format of the output from the operator commands is described in the *ExHPDM Messages and Codes* manual.

See the chapter “About This Book” for the syntax conventions used in this chapter. Note that parenthesis are optional in all operator command; however, they are shown in the syntax diagrams and examples in the interest of precision.

An example of coding a command with parenthesis, and the equivalent command coded without parenthesis, and a more readable format, is shown here:

```
SOV DISPLAY STREAM(ALL)
SOV DISPLAY STREAM ALL
```

Parenthesis may sometimes be required to ensure that a command is not ambiguous. An ambiguous command may result if a keyword value happened to be the same as a keyword. For example, if a client job name happened to be called DETAIL, then the following command would require parenthesis:

```
SOV DISPLAY CLIENT DETAIL
```

In this example, parenthesis are required because DETAIL is a keyword. The minimum parenthesis required would be:

```
SOV DISPLAY CLIENT(DETAIL)
```

CANCEL Commands

The CANCEL commands allows cancellation of particular client requests that are active or waiting and cancellation of stream connections.

Cancelling connection(s) results in an I/O error being generated for the client(s) and an appropriate message issued to the client(s) JESYSMSG. See “Abnormal Termination of Client Connections” on page 45 for details.

Syntax

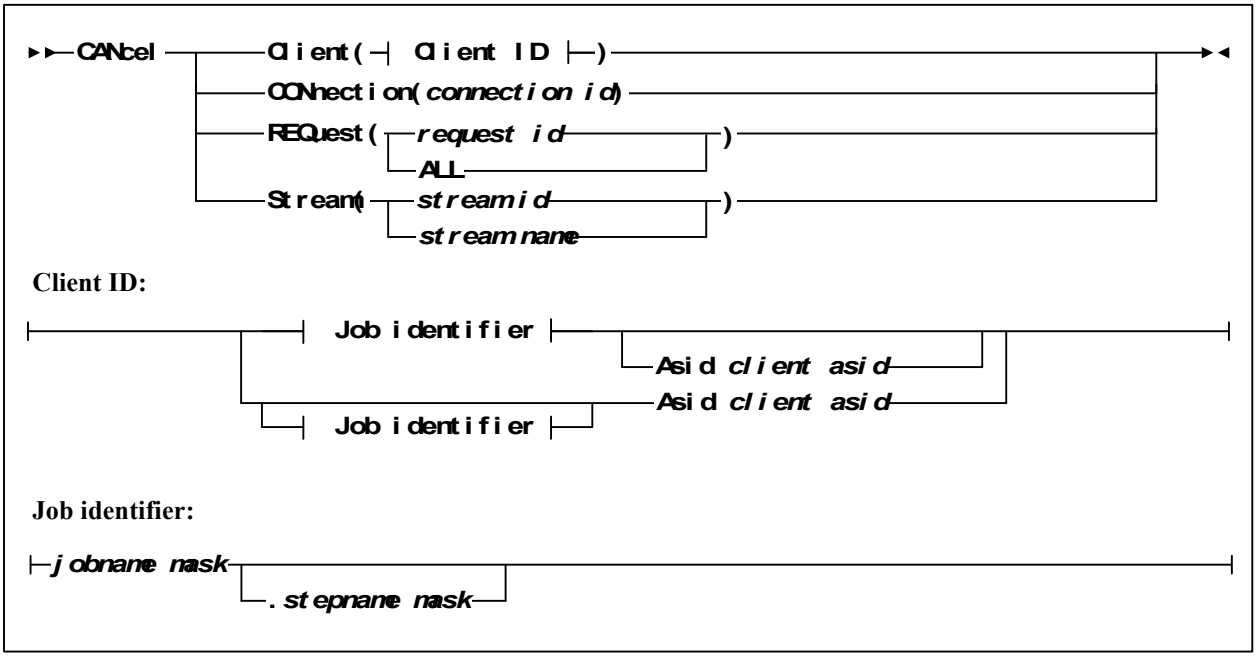


Figure 30. CANCEL Command

Command Name

CANcel
initiates the CANCEL command.

Parameters

Stream
specifies that all connections for an instance of a stream (*stream id*) or all streams of a particular name (*stream name*) are to be cancelled. The *stream id* and *stream name* can be obtained through the use of the DISPLAY STREAM ALL operator command (e.g., D S ALL).

Client
specifies that all connections for an instance of a client are to be cancelled. The Client ID can be obtained through the use of the DISPLAY CLIENT command (e.g,. D C *) and may be specified as a Job Identifier and/or Address Space ID.

If all clients belonging to a particular stream are cancelled, this does not result in the stream being cancelled. If a RETAIN period is set for the stream, the stream enters its retain processing. If a RETAIN period is not specified, the stream terminates unless there are outstanding waiting connections to process. To cancel the stream, issue a CANCEL STREAM command.

Job Identifier

The client job identifier is expressed in the format *jobname mask.stepname mask*. The fullstop is used to separate these fields. *stepname mask* is optional.

ExHPDM masking characters may be used in either of the names as described in Table 4. on page 138.

The client jobname and stepname may be obtained through the use of the DISPLAY CLIENT command (e.g., D C *).

Asid

asid specifies the client Address Space ID. The ***asid*** can be obtained through the use of the DISPLAY CLIENT command (e.g., D C *).

CONnection

connection id specifies that an individual connection is to be cancelled. The *connection id* can be obtained through the use of the DISPLAY CLIENT command (D C *) or the DISPLAY STREAM command (e.g., D S ALL) command.

REQuest

request id specifies that an individual request is to be cancelled. The *request id* can be obtained through the use of the DISPLAY REQUEST DETail command (D REQ DET). All requests may be cancelled by specifying ALL (e.g. CAN REQ ALL).

Cancel Example

This example cancels the ExHPDM stream tasks started with the name DB2.

```
SOV CAN S(DB2)

SOV06401I All tasks for stream DB2 cancelled.
SOV06043I Stream DB2 task 35 has been terminated.
```

DISPLAY Commands

The ExHPDM DISPLAY commands allow you to monitor the status of the server and its clients.

Syntax

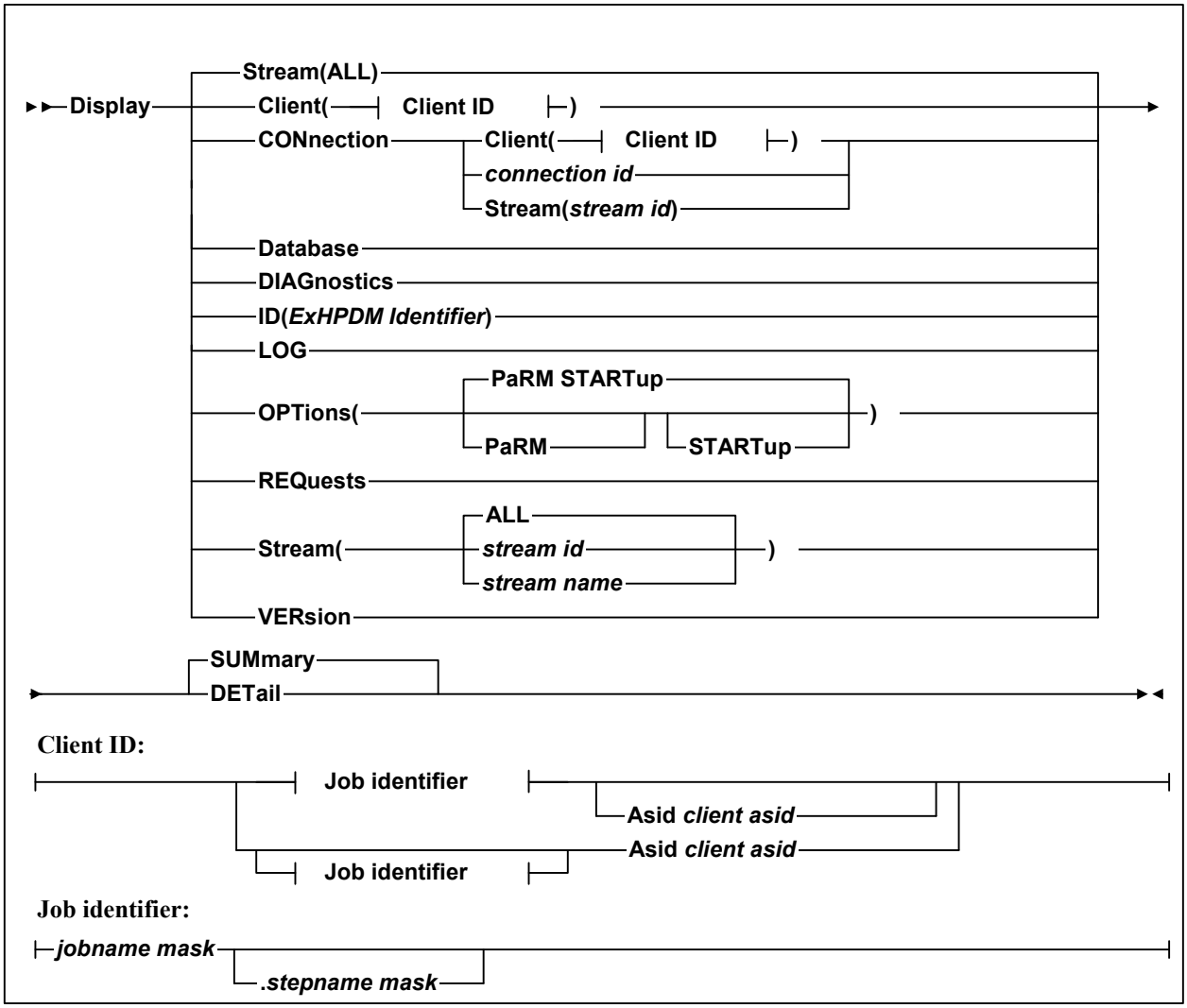


Figure 31. DISPLAY Commands

Command Name

Display
initiates the DISPLAY command

Parameters

Client
specifies information for an instance of a client or clients are to be displayed. The Client ID may be specified as a Job Identifier and/or Address Space ID.

Job Identifier

The client job identifier is expressed in the format *jobname mask.stepname mask*. The fullstop is used to separate these fields. *stepname mask* is optional.

ExHPDM masking characters may be used in either of the names as described in "ExHPDM Masking Characters" in Table 4. on page 138.

Asid

asid specifies the client Address Space ID.

CONnection

specifies that connection specific information for a particular *connection id*, *stream id*, or Client ID is to be displayed.

Stream

stream id specifies that the connections for this stream are to be selected.

Client

specifies that the connection for Client ID are to be displayed. The Client ID may be specified as a Job Identifier and/or Address Space ID as per DISPLAY CLIENT. See DISPLAY CLIENT for details.

DataBase

indicates that database-related information is to be reported. Includes only the utilization values.

DIAGnostics

specifies ExHPDM diagnostic information is to be displayed. DIAGnostic allows ExHPDM technical support to diagnose internal ExHPDM problems.

ID

specifies information for the *ExHPDM Identifier* is to be displayed. The two types of identifiers are *stream id* and *connection id*. All the identifiers used with ExHPDM are unique. The appropriate display is output according to the type this ID belongs to. For example, if the specified *ExHPDM Identifier* belongs to a stream, the display details are the same as DISPLAY STREAM.

LOG

displays the most recent 120 output lines of the ExHPDM log.

OPTions

specifies the options from the SOVPRM*xx* member and/or the startup should be displayed.

PaRM

specifies the option information for the SOVPRM*xx* parameter file.

STARTup

specifies the option information for the ExHPDM startup.

PERFormance
specifies the instantaneous flow rate based on moving and weighted averages over the past second, ten seconds, and two minutes. For time intervals greater than one second, this parameter also shows maximum and minimum rates over any one second in the interval and variance in the one-second samples.

ExHPDM keeps track of I/O performance in one-second minimum elapsed time measurement intervals. The raw data for each interval is the number of blocks transferred, converted to KB by multiplying by the block size in KB. The most recent 120 samples are kept in a circular table in storage. See Display Example 11.

STREAM PERF, CLIENT PERF, and CONNECTION PERF commands display similar output.

REQuests
specifies information for current and waiting requests is to be displayed. A request is a unit of work that is given to ExHPDM to perform.

Stream
specifies stream information for selected stream tasks, with a particular *stream id* or *stream name*, or ALL stream tasks is to be displayed.

STREAM(ALL) is the overall DISPLAY default.

VERsion
displays the version, release and modification number of the ExHPDM server, the version of the common parser used by ExHPDM, and the license key information.

SUMmary
specifies summary information is required for the DISPLAY command.

DETail
specifies detailed information is required for the DISPLAY command.

DISPLAY Examples

Example 1: Summary Display of All Clients

The following example is a summary display of all clients.

```
SOV D C(*)

SOV02103I ExHPDM command issued.

SOV06068I
Client                    ASID    Identifiers    Status
     Connection Stream
------------------------------ ----  ---------- ------  -------
JOB:OWPLFDR.DUMP            0017       66    67  Active
JOB:OWPLFDR.DUMP            0017       68    67  Active

SOV06067I
Total    Connections : 2
Selected Connections : 2
```

Example 2: Detailed Display of All Client Connections

The following example is a detailed display of all client connections. Note that the optional parentheses on the command are being omitted in this example.

```
SOV D CON C * DET

SOV02103I ExHPDM command issued.

SOV06059I
Connection    Client    Stream    Flow Rate    R/W
  ID          ASID      ID        KB/sec
----------  ----------  ----------  -----------  ---
      66       0017        67        1771   W
      68       0017        67        1733   W

                    MVSR  SOV06058I
                    Total Connections:    2
                    Selected Connections: 2
```

Example 3: Detailed Display of all Requests

The following example is a detailed display of all requests. Note that the Display command itself is also shown in this output as it is also a request (OPERCMD).

SOV D REQUESTS DET

SOV02103I ExHPDM command issued.

SOV06050I Total requests: Active 3, Waiting 0
SOV06051I CONNECT:
Req ID Status Request
------ ------ --
66 Active OWPL.BACKUP.VVTPE01.FDR
68 Active OWPL.BACKUP.VVTPE02.FDR

SOV06051I OPERCMD:
Req ID Status Request
------ ------ --
73 Active OPERCMD D REQUESTS DETAIL

Example 4: Summary Display of All Stream Tasks

The following example causes a summary display of all stream tasks.

```
SOV D S

SOV02103I ExHPDM command issued.
SOV06054I
Stream              Tasks     Connections
------------        -------   -----------------
TLDSSTRM                 0           0
TLSTRM                   0           0
DBASTRM                  0           0
REDST3A                  0           0
REDST3B                  0           0
REDST3C                  0           0
SOV06053I
Total    stream tasks :  2 Connections : 7
Selected stream tasks :  2 Connections : 7
```

Example 5: Detailed Display of Active Stream Tasks

The following example creates a detailed display of active stream tasks for stream STRM4490. Note that stream STRM4490 with ID 25 is currently waiting for an Operator reply to an outstanding WTOR. This is indicated in the Status field.

```
SOV D S(STRM4490) DET

SOV02103I ExHPDM command issued.

SOV06055I Stream : STRM4490 ID : 43  Device : 0B80  VOL : 542628
File : IMSDEPT.DTUE.NIGHT.BACKUP.B72M1FN3
Write Flow Rate : 0 KB/sec Blocks : 1152
Status : Ready
Volumes : Current 1, NNCA 40, MAX 50
Elapse Time (HH:MM:SS) : 00:02:13
Client  Client   Client   Connection Class    Blocks   Elapse Time
ASID    Name     DD       ID                  HH:MM:SS
------- -------- -------- ---------- --------- -------- -----------
0028    DBAPRROD OWPL     46       DBAPROD    860      00:02:09

SOV06055I Stream : STRM4490 ID : 25  Device :       VOL :
File : APPLIED.PHYSICS.RDDEPT.WEEKLY.BKUP.B72JG681
Write Flow Rate : 0 KB/sec Blocks : 0
Status : Operator reply
Volumes : Current 0, NNCA 40, MAX 50
Elapse Time (HH:MM:SS) : 00:13:03
Client  Client   Client   Connection Class    Blocks   Elapse Time
ASID    Name     DD       ID                  HH:MM:SS
------- -------- -------- ---------- --------- -------- -----------
0025    DBAPROD4 OWAS     24       REDCL3B    0       00:13:04
0025    DBAPROD4 OWPL     26       REDCL3B    0       00:13:02
0025    DBAPROD4 OWCH     27       REDCL3B    0       00:13:01
0025    DBAPROD4 OWPT     28       REDCL3B    0       00:13:00
SOV06053I
Total    stream tasks : 2 Connections : 5
Selected stream tasks : 2 Connections : 5
```

Example 6: Detailed Display of Client DBAPRPD4

The following example creates a detailed display of client DBAPROD4.

```
SOV D C(DBAPROD4) DET

SOV02103I  ExHPDM command issued.

SOV06069I  Client : JOB:DBAPROD4.DBAPLDSN  ASID : 0028
DD name  Data Set Name                    Identifier
                                Connection  Stream
-------  --------------------------------  ----------- ------
OWAS     ZAYNARD.RD3B.FULL.KUP1               65     66

SOV06069I Client : JOB:DBAPROD4.DBAPLDSN  ASID : 0028
DD name  Data Set Name                    Identifier
                                Connection  Stream
-------  --------------------------------  ----------- ------
OWPL     ZAYNARD.RD3B.FULL.SEC1C4             67     66

SOV06069I Client : JOB:DBAPROD4.DBAPLDSN  ASID : 0028
DD name  Data Set Name                    Identifier
                                Connection  Stream
-------  --------------------------------  ----------- ------
OWCH     ZAYNARD.RD3B.FULL.KU1C2              68     66

SOV06069I Client : JOB:DBAPROD4.DBAPLDSN  ASID : 0028
DD name  Data Set Name                    Identifier
                                Connection  Stream
-------  --------------------------------  ----------- ------
OWPT     ZAYNARD.RD3B.FULL.KU1C3              69     66
SOV06067I
Total    Connections : 4
Selected Connections : 4
```

Example 7: Detailed Display of Connections for Client DBAPROD4

The following example creates a detailed display of connections for client DBAPROD4.

```
SOV D CON CLIENT DBAPROD4 DET

SOV02103I  ExHPDM command issued.

SOV06059I
Connection    Client     Stream     Flow Rate       R/W
   ID         ASID        ID        KB/sec
---------    ------    --------   -----------    ----
     65       0028         66            0        W
     67       0028         66            0        W
     68       0028         66            0        W
     69       0028         66            0        W
SOV06058I
Total    Connections : 4
Selected Connections : 4
```

Example 8: Original, Effective, and Change Parameter Values for ExHPDM Server

The following example displays the Original, Effective and Change parameter values for the ExHPDM server. The startup parameter are excluded as only the PRM option is specified. Note that the LOGFILE is currently being changed from SYSOUT(E) to DSN(OWPL.LOG). This is indicated by the Change value setting.

```
SOV D OPT PRM
SRV02103I ExHPDM command issued.

SRV06900I Parameter file is DSN:OWPL.STKPARM(SOVPRMPL) 977
Parameter   (O)riginal Parameter file value
        (E)ffective value
        (C)hange value ; in progress
------------ - ------------------------------------------------
DATABASE     O EXHPDM.OWPL.DBASE3
BACKDSN      O Not specified
JOURNAL      O Not specified
BACKJRNL     O Not specified
MONITOR      O DATABASE THRESH(80%) WARN INTERVAL(5M )
         E DATABASE THRESH(80%) WARN INTERVAL(5M )
         E JOURNAL THRESH(80%) WARN INTERVAL(5M )
PREFIX(MSG)  O SRV
         E SRV
PREFIX(CMD)  O Not specified
         E SOVP
SAF        O Not specified
         E FACILITY
LOGFILE      O SYSOUT(E)
         E SYSOUT(E)
         C DSN(OWPL.LOG)
TMS        O CONTROLT
         O SCRVOLRTN(SOVCS399)
         O SSNAME(HPDM)
         O DMNAME(EXHPDMDM)
         E CONTROLT
         E SCRVOLRTN(SOVCS399)
         E SSNAME(HPDM)
         E DMNAME(EXHPDMDM)
REQUEST      O Not specified
         E ADMIN     : MAXLIMIT(20) LOG(NO)
         E CONNECT   : MAXLIMIT(200) LOG(NO)
         E OPERCMD   : MAXLIMIT(20) LOG(NO)
         E VALIDATE  : MAXLIMIT(200) LOG(NO)
SMF                  O Not Specified
                     E TYPE(196) SYNC (YES)
                     E INTERVAL (NOT SPECIFIED) OFFSET (NOT SPECIFIED)
ACTIVATE         O Not Specified
         E Not Specified
```

Example 9: Original and Effective Startup Values for ExHPDM Server

The following example displays the Original and Effective startup values for the ExHPDM server. The parameter file values are excluded as only the START option is specified:

```
SOV D OPT START

SRV02103I ExHPDM command issued.
SRV06905I Start up parameters.
Parameter   (O)riginal start value
          (E)ffective value
------------ - ------------------------------------------------------------
PRM          O PL
           E PL
MEMBER       O Not specified
           E SOVPRM
QUIET        O Not specified
           E ON
VERBOSE      O Not specified
           E OFF
TRACE        O Not specified
           E OFF EID(0) FID(0)
SSNAME       O SOVP
           E SOVP
KEY          O Not specified
           E 5
```

Example 10: DISPLAY STREAM DETAIL Output

The DISPLAY STREAM DETAIL command generates the performance output shown in the following example.

```
- MVSR SOV06055I Stream : STREAM ID : 18  Device : OB81  VOL : 542497
- File : OWCH.STREAM1 MXUHUHJ1
- Write Flow Rate : 0 KB/sec Blocks : 68
- Separation: Jobname (OWCHMA  )
- Instantaneous Flow Rate : 256 KB/sec
- Status : Ready
- Volumes : Current 1, NNCA 40, Max 50
- Elapse Time (HH:MM:SS): 00: 01: 19
- Client  Client    Client    Connection  Class    Blocks  Elapse Time
-  ASID    Name       DD          ID                  HH:MM:SS
- ------- -------- --------    --------   --------  ------  -----------
-    0017 OWCHMA    SUBSYSDD        17   Class 1       68   00:01:19
- MVSR  SOV06053I
- Total    stream tasks : 1 Connections : 1
- Selected stream tasks : 1 Connections : 1
```

Example 11: DISPLAY STREAM PERF

The following example shows results derived from the table which stores the most recent 120 samples of I/O performance. Similar displays are output for DISPLAY STREAM DETAIL, DISPLAY CLIENT DETAIL, AND DISPLAY CONNECTION DETAIL operator commands.

```
- MVSR SOV06095I Stream : STREAM1 ID : 35  Device : 23686C  VOL : 542561
-          Meas. interval ->          1s          10s          120s
-                   Avg kB/s         256          435           448
-                   Max kB/s           -          512           512
-                   Min kB/s           -          256           256
-          Variance kB/s               -        15485         13405
- MVSR SOV06053I
- Total          stream tasks :      1 Connections :      1
- Selected       stream tasks :      1 Connections :      1
```

Example 12: DISPLAY CLIENT DETAIL Output

The DISPLAY CLIENT DETAIL command generates the following output.

```
- MVSR      SOV06069I Client : JOB:OWCHMA.DUMPDSN  ASID : 0017
- Instantaneous Flow Rate: KB/sec
- DD name         Data Set Name                                  Identifiers
-                                                                Connect        Stream
- -------         -------------------------------------          -------        -------
- SUBSYSDD        OWCH.SYS3.DUMP                                   17             18
- MVSR  SOV06067I
- Total            Connections : 1
- Selected         Connections : 1
```

Example 13: DISPLAY CONNECTION DETAIL Output

The following output is generated as a result of a DISPLAY CONNECTION DETAIL command.

```
- MVSR  SOV06059I
- Connection          Client      Stream          Flow rate      R/W      IFR
-       ID             ASID          ID             KB/sec                KB/Sec
- -------------       --------   ------------    -----------    ------   ----------
-          17          0017           18            183          W        1536
- MVSR  SOV06058I
- Total          Connections : 1
- Selected       Connections : 1
```

Example 14: Command to Display Database Utilization

The following output is generated as a result of a DISPLAY DATABASE command.

```
SOV06243I Database Name: VTAPE.SS0V300.DATABASE 763
     Percentage full : 11%
           Status : OK
      Last Backup : 01SEP2003
    Last Backup DSN : VTAPE.SS0V300.DATABASE.BACKUP.V0000033
      Backup Limit : 3
     Journal  name : VTAPE.SS0V300.JOURNAL
    Percentage full : 3%
     Journal Status : OK
    Last Backup DSN : VTAPE.SS0V300.JOURNAL.BACKUP.V0000033
```

Example 15: Command to Display the ExHPDM version information

The following output is generated as a result of the DISPLAY VERSION command.

```
SOV069401 ExHPDM Version is 6.2.0
SOVO6941I ExHPDM is utilizing the facilities of the command parser version 2.5.0
 SOVO6942I ExHPDM License Key Information
       Type:         Permanent
       Customer:     Test
       Site:         999999
       Expiry Date:  2006365
       Serial Number:  123456
```

SET Command

The ExHPDM SET command allows options and parameters to be set up and changed while ExHPDM is active. Any options changed are reflected as the new effective values displayed by the Display OPTions command.

Syntax

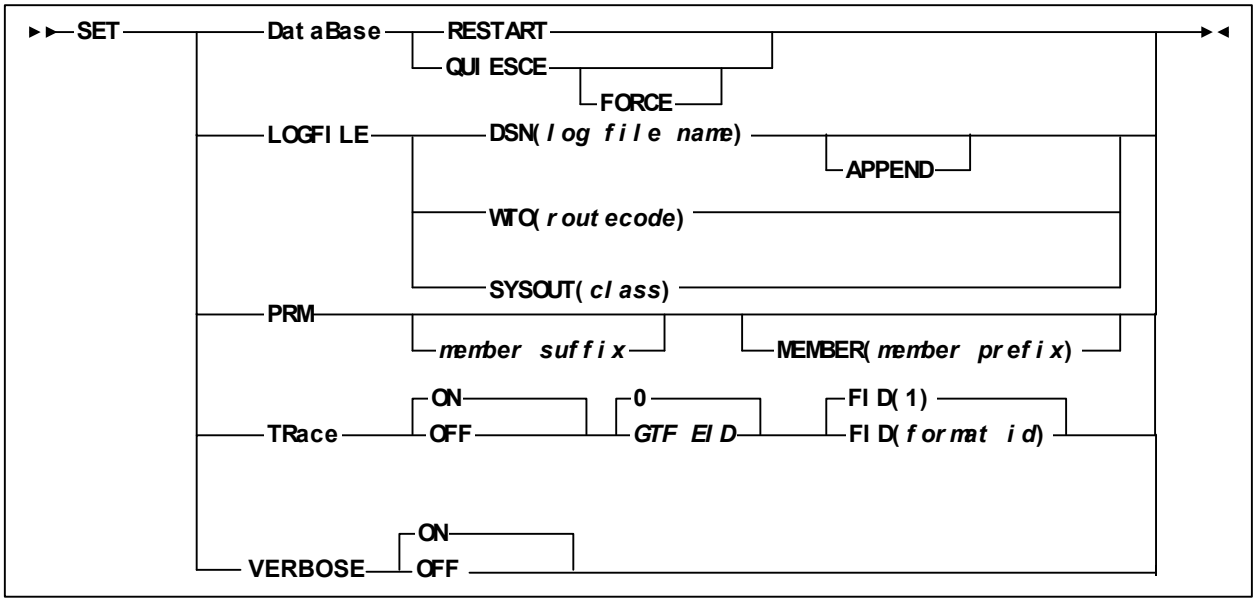


Figure 32. SET Command

Command Name

SET
initiates the SET command.

Parameters

DataBase
allows for the ExHPDM database recovery processing. If the ExHPDM server is started in disaster recovery mode, the **DataBase** parameter is not supported.

RESTART
allows access to the database to resume after being quiesced. If the ExHPDM server is started in disaster recovery mode, the **RESTART** parameter is not supported.

QUIESCE
stops access to the database when all existing clients and stream tasks have completed processing. That is, the database is quiesced. No connection requests to ExHPDM are allowed.

FORCE

The **FORCE** parameter can be used to immediately cancel all stream and client connections.

If the ExHPDM server is started in disaster recovery mode, the **QUIESCE** parameter is not supported.

To resume access to the database issue the **SET DB RESTART** command.

LOGFILE

causes ExHPDM to switch the log output to the destination as follows:

DSN

log file name specifies the MVS data set name. This file is allocated and cataloged if it does not currently exist. If APPEND is not specified and the file exists, the file is over written. If an error occurs while writing to the data set then the log output is redirected to the MVS SYSLOG.

APPEND

specifies that the log file output is to be appended to the file that currently exists and has the same file name.

WTO

routecode specifies the console route code. The valid values are 0 to 128. WTO(0) indicates that log output is directed to the MVS SYSLOG.

SYSOUT

sysout_class is any valid JES sysout class.

PRM

specifies that the ExHPDM parameter file should be reread from SYS1.PARMLIB or the STKPARMS DD. The DATABASE *DB file name,* the DATABASE JOURNAL *journal file name,* and REQUEST parameters cannot be changed using this command. ExHPDM needs to be restarted for these to take effect.

If the ExHPDM server is started in disaster recovery mode, the PRM parameter is not supported.

Note: If SET PRM is issued and there are currently active streams tasks, then any changes in parameter values that affect stream tasks, for example changes to DEVICE and STREAM, do not take effect until the stream tasks have ended.

If SET PRM is issued without the *member suffix* and/or *member prefix,* the last values for these are used.

member suffix

specifies the optional two character suffix for the parameter member name. The member name is formed using the *member prefix* specified in the MEMBER parameter. If the STKPARMS DD is a sequential data set (non PDS), the PRM suffix is ignored. However, the parameters are still

reread. The default PRM value is that which is used in the startup specification of ExHPDM or from the previous settings from a SET command. The Display OPTions command can be used to examine the current PRM value.

MEMBER

member prefix specifies the 1 to 6 character parameter member name prefix. The member name is formed using the suffix specified in PRM. If STKPARMS DD is a sequential data set (non PDS), the MEMBER prefix is ignored. However, the parameters are still reread. The default MEMBER value is that which is used in the startup specifications of ExHPDM or from the previous setting from a SET command. The Display OPTions command can be used to examine the current MEMBER value. -

TRace

turns GTF tracing ON or OFF. GTF must be active for tracing TYPE=USR events. GTF tracing of ExHPDM events is automatically turned off when ExHPDM is shutdown.

If SET TR is specified the default is to turn tracing ON**.**

trace id (GTF EID)

trace id specifies that tracing only occurs to GTF where this matches the event identifier (EID) on the USRP GTF keyword for the currently active GTF trace. This may be required to avoid tracing other nonExHPDM related GTF trace records. A value in the range x‘000’ to x‘3FF’ should be specified.

If USRP is not currently in effect for the TYPE=USR GTF trace, tracing always occurs. The selected *trace id* is in effect until another SET TRACE command is entered to change the value.

For addition details about using GTF and the USRP keyword, refer to the *MVS/ESA SP V5 Diagnosis: Tools and Service Aids.*

The default *trace id* is 0.

Note: as the values specified on the GTF USRP are specified as hexadecimal values it may be easier to use hex values for the *trace id*. A hexadecimal value is entered as x’*nnn*’. Where *nnn* matches the GTF EID.

FID

format id specifies the trace formatting appendage to use when formatting the GTF trace entries with IPCS. The *format id* may be any value from decimal 0 to 80. The *format id* indicates which AMDUSR*nn* formatting routine is used by the IPCS GTFTRACE command. Where *nn* is the *format id* and is in the range hexadecimal 00 to hexadecimal 50 (decimal 80). The selected *format id* is in effect until a SET TRACE command is entered to change the value.

The default format id is 1.

Note: As the value to determine the AMDUSR*nn* formatting routine is specified as a hexadecimal value, it may be easier to use hex values for the *format id*. A hexadecimal value is entered as x'*nn*'.

VERBOSE
turns VERBOSE message output ON or OFF. One of these must be specified. ExHPDM messages are only output when log file messages are being directed to a data set or to SYSOUT. No VERBOSE messages are output when the LOGFILE WTO option is specified. This command should only be entered when requested by StorageTek Support.

SET Examples

SET TRACE Examples

Example 1

The following example initiates a GTF trace of ExHPDM.

```
SOV SET TRACE ON
```

Example 2

The following example turns off a GTF trace of ExHPDM.

```
SOV SET TRACE OFF
```

Example 3

The following example turns off a GTF trace of ExHPDM and changes the current trace id. This is used next time the trace is turned on.

```
SOV SET TRACE OFF 25
```

Example 4

The following example turns on a GTF trace of ExHPDM and changes the current trace id. A formatting appendage AMDUSR02 is required at IPCS GTFTRACE time.

```
SOV SET TRACE ON 32 FID(02)
```

SET LOGFILE Examples

The following examples show the various **SET LOGFILE** commands and the output from those commands.

Example 1

The following example logs ExHPDM log messages to the MVS SYSLOG.

```
SOV SET LOGFILE WTO(0)
```

Example 2

The following example allocates a new ExHPDM log data set.

```
SOV SET LOGFILE DSN(USER1.LOG)
```

Example 3

The following example appends to an existing ExHPDM data set. Note that the optional parenthesis have not been used.

```
SOV SET LOGFILE DSN USER1.OLDLOG APPEND
```

Example 4

The following example directs ExHPDM log messages to SYSOUT.

```
SOV SET LOGFILE SYSOUT(H)
```

Example 5

The following example allows the OUTLIM value to be specified on the LOGFILE keyword in the startup parameter file and on the SET command.

Note: The parentheses are not required.

```
SET LOGFILE . . .
        SYSOUT (sysout_class [OUTLIM(line_limit)])
```

SHUTDOWN Commands

The ExHPDM SHUTDOWN commands allow the ExHPDM server to be stopped.

Issuing a SHUTDOWN command stops any further connections from being accepted from clients. If any requests are still active, the IMMEDIATE parameter can be used to cancel the requests. Each shutdown type is hierarchical, that is, the order of shutdown can only be entered in the order with the NORMAL parameter first followed by the IMMEDIATE parameter.

The **IMMEDIATE** parameter notifies any current active connections and stream tasks to complete immediately. This results in an I/O error being generated for active client connections and an appropriate message issued to the clients JESYSMSG. See “Abnormal Termination of Client Connections” on page 45 for details.

If any connections are still active and the **IMMEDIATE** parameter is not specified, the ExHPDM server shuts down when all current requests have completed.

Syntax

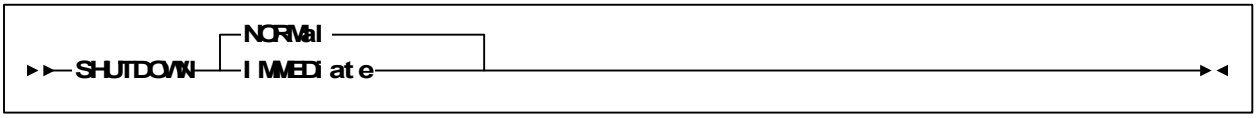


Figure 33. SHUTDOWN Command

Command Name

SHUTDOWN
initiates the SHUTDOWN command.

Parameters

NORMAL
specifies that ExHPDM is to shutdown if no connections are in process. ExHPDM waits for all connections to close prior to shutdown. This parameter is the default.

IMMEDiate
specifies that ExHPDM is to shutdown connections that are in progress. The **NORMAL** parameter must be specified before the **IMMEDiate** parameter is specified. Otherwise the immediate shutdown is changed to a normal shutdown and the **IMMEDiate** parameter must be specified again for an immediate shutdown.

SHUTDOWN Command Examples

The following examples show the various SHUTDOWN commands and the output from these commands.

Example 1

This example shows a SHUTDOWN command being issued when there are still active client connections.

```
SOV SHUTDOWN
SOV02103I ExHPDM command issued.
SOV06202I SHUTDOWN NORMAL issued.
SOV06007I SHUTDOWN command from CN:OPSO2 accepted.
SOV03104W There are active streams or requests.
SOV06054I
StreamTasksConnections
_____________________________________________________

TLDSSTRM00
TLSTRM00
DBASTRM12
REDST3A00
REDST3B00
REDST3C00
SOV06053I
Total    stream tasks : 1  Connections : 2
Selected stream tasks : 1  Connections : 2

SOV06050I Total requests: Active 3, Waiting 0
SOV0652I
Request TypeActiveWaiting
_____________________________________________

ADMIN00
CONNECT20
OPERCMD10
VALIDATE00
```

Example 2

This example stops the ExHPDM server when there is an active stream task.

```
SOV SHUTDOWN IMMEDIATE
SOV02103I ExHPDM command issued.
SOV06202I SHUTDOWN IMMEDIATE issued.
SOV06007I SHUTDOWN command from CN:OPSO2 accepted.
SOV06043I Stream DBASTRM task 50 has been terminated.
SOV03200I ExHPDM has shut down.
```

START Command

The ExHPDM START command allows new stream tasks for write operations to be started. This allows streams to be ready for connections from clients prior to clients being started.

Note that the limit for any named stream being started is defined in the STREAM keyword by the CONCURRENT parameter in the SOVPRM*xx* startup parameter file.

Syntax

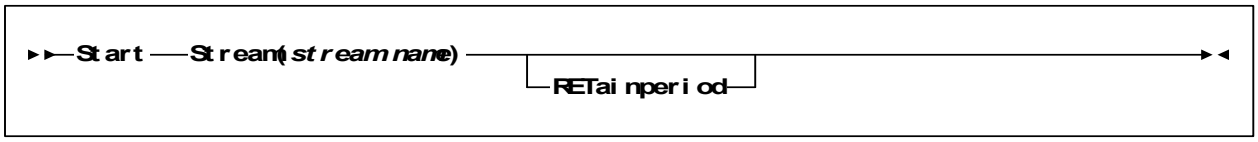


Figure 34. START Command

Command Name

Start
initiates the START commands.

Parameters

Stream
stream name specifies the name of the stream task to start. Note that the limits specified by the **CONCURRENT** parameter in the STREAM keyword is applied to the newly started task. The start is rejected if this limit is exceeded.

RETainperiod
specifies that the time period of the DEVICE **RETAIN** parameter should commence immediately from the time the stream is started. Note that if the DEVICE **RETAIN** period is 0, the stream task does not start. A message is displayed indicating this condition.

If RETainperiod is not specified, the stream task only commences with the retain period when all client connections have closed.

START Examples

Example 1

The following example starts a stream task for the stream definition DB2.

```
SOV S S(TLSTRM)
SOV02103I ExHPDM command issued.
*IEC501A M OB81,PRIVAT,SL,COMP,SOVWT,SOVWT,
OPERDEPT.MSVR.SOVWT.HPDMETL.STREAM.B7329UG2
IEC705I TAPE ON OB81,542630,SL,COMP,SOVWT,SOVWT,
OPERDEPT.MSVR.SOVWT.HPDMETL.STREAM.B7329UG2
SOV06211I TLSTRM write stream task (ID 48) started.
```

Example 2

The following example starts a stream task for the stream definition TLSTRM using the RETAINPERIOD parameter. Note that the retain period for the selected stream device is 0.

```
S STREAM(TLSTRM) RETAINPERIOD
SOV02103I ExHPDM command issued.
SOV06210E TLSTRM could not be started. Retain period for device TLDEV is 0.
SOV06214I TLSTRM stream task (ID 57) ended.
```

Chapter 9. ExHPDM Administration Utility (SOVADMN)

Overview

The ExHPDM Administration Utility, SOVADMN, is used in batch or dynamically invoked from an application program to perform the following functions:

- Database processing to:
    - List database elements
    - Update database elements
    - Delete database elements
    - Restore backup data set to current database
    - Backup database and journal data set for current database
    - Process stream and client file expiration.
- Stream tape volume processing for ExHPDM stream tape data sets when the ExHPDM database is inaccessible or not available. Such processing is required in a disaster recovery situation.

Running SOVADMN as a Batch Job

Specifying an ExHPDM Server

The ExHPDM administration utility is only available when the ExHPDM server is active in the current MVS. Connection to the server is specified through either the JCL parm SSNAME value or the //SNAM*nnnn* DD DUMMY statement, where *nnnn* is the ExHPDM subsystem name. If SSNAME is specified, then any //SNAM*nnnn* DD DUMMY specifications are not used.

Multiple //SNAM*nnnn* DD DUMMY statements can be given. The first active ExHPDM system found in the list is the one that is used. For example:

```
//SNAMSOV1 DD DUMMY
//SNAMSOV2 DD DUMMY
//SNAMSOV3 DD DUMMY
```

If SOV1 is active, it is used; otherwise SOV2, if it is active; if not, SOV3.

JCL Requirements

The user selects ExHPDM actions by means of utility commands specified through the SYSIN file. The commands are categorized by:

- DATABASE LIST / UPDATE / DELETE for database processing
- DATABASE BACKUP / RESTORE
- SCANSTReamfile for processing in data recovery mode
- SCANEXPIRE for stream and client file expiration.

Any of these can be invoked once or more in the SOVADMN job.

See “About This Book” for the syntax conventions used in this chapter. Note that parenthesis are optional on all syntax diagrams in this chapter. Refer to “SYSIN DD Requirements” on page 215 for general information on using parentheses in SOVADMN commands.

The SOVADMN utility accepts the following options in the JCL EXEC parameter.

Syntax

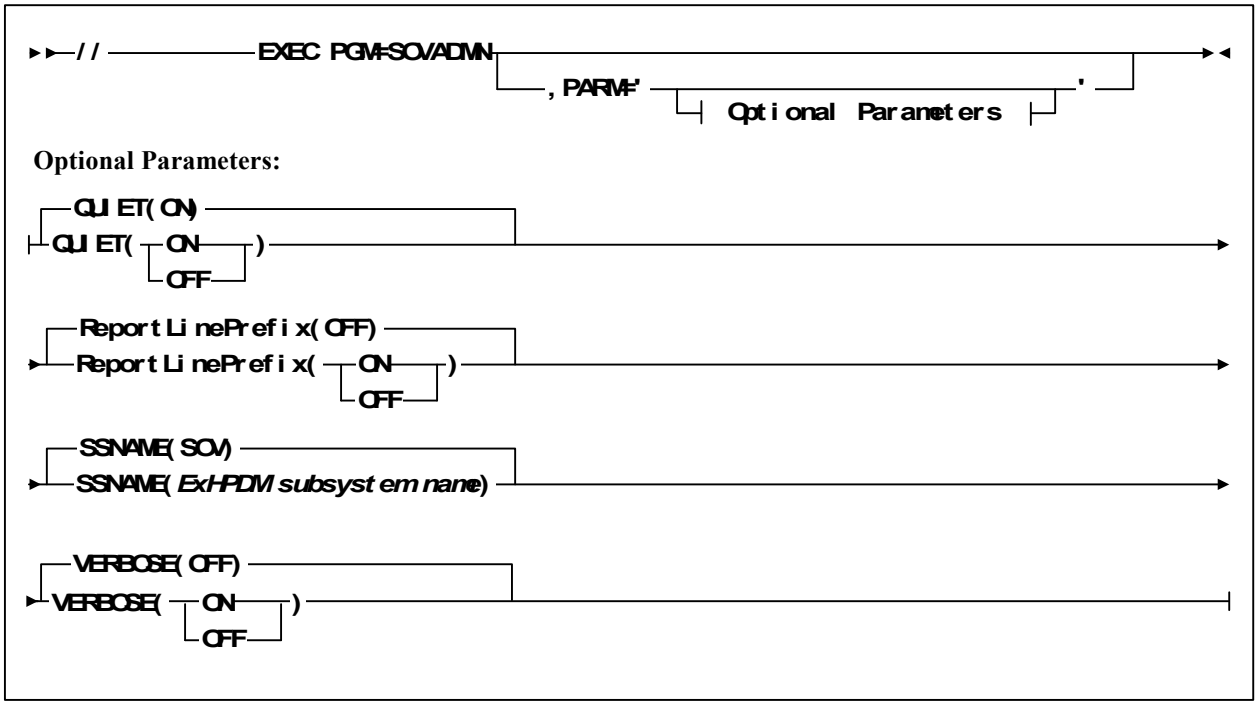


Figure 35. SOVADMN Utility

Utility Name

SOVADMN is the name of the program to execute (EXEC PGM=SOVADMN).

Parameters

The following options are coded in the JCL EXEC parameters.

SSNAME

ExHPDM subsystem name indicates the subsystem name of the ExHPDM server that SOVADMN will use. All of the ADMIN directives are directed to this server. This value overrides the SNAM*nnnn* DD DUMMY specifications.

SSNAME(SOV) is the default.

ReportLinePrefix

specifies whether or not report line prefixes are required. Report line prefixes are available on each report line to assist programmatic post processing of the report file.

OFF

Do not generate any report line prefixes.

ON

Generate report line prefixes. The following types of prefixes are used:

C: Client Information

H: Header Text—Common

I: Informational Text

W: Warning Text

E: Error Text

J: List of Journals

L: List of volumes

S: Stream Information

T: Total

U: Unit Information

V: Volume Information

+: Continuation

ReportLinePrefix(OFF) is the default.

VERBOSE

This parameter is only to be used when advised to by StorageTek Technical Support. VERBOSE(ON) specifies that internal messages are to be output. Note that the use of this parameter might degrade performance.

VERBOSE(OFF) is the default.

QUIET

This parameter is only to be used when advised to by StorageTek Technical Support. QUIET(OFF) specifies that internal library diagnostic messages are to be output.

QUIET(ON) is the default.

JCL Examples

Example 1

The following example shows SOVADMN being invoked by standard JCL to connect to the ExHPDM server whose MVS subsystem name defaulted to SOV:

```
//ADMINJOB  JOB jobcard info
//UTIL      EXEC PGM=SOVADMN,PARM='SSNAME(SOV)'
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
ADMIN DB LIST
//
```

Example 2

In the following example, the presence of the JCL DD statement SNAMSOVW causes the utility to connect to the server whose MVS subsystem is called SOVW:

```
//ADMINJOB  JOB jobcard info
//UTIL      EXEC PGM=SOVADMN,PARM='RLP(ON)'
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT  DD SYSOUT=*
//SNAMSOVW  DD DUMMY          Use this ExHPDM server
//SYSIN     DD *
ADMIN SCANEXPIRE
ADMIN DB BACKUP
//
```

SYSIN DD Requirements

The commands to the SOVADMN utility are provided in the SYSIN DD. This DD is scanned for lines starting with the keyword ADMIN. The command to be executed is comprised of this line and the following lines up to the next ADMIN keyword. More than one ADMIN command may be specified. The commands are executed in sequence, with the output from each command sent to the SYSPRINT DD in the same order as specified in SYSIN.

Warning: The SOVADMN SYSIN DD does not support line numbers.

When coding more than one ADMIN command in the same SYSIN DD, the ADMIN keyword must be the first word on the line in which it appears.

The SYSIN DD may be in fixed or variable length format, with record length up to 255. The SYSPRINT DD should have fixed or variable length record format, with a minimum record length of 132.

When coding the commands in SYSIN, space may be used freely to delimit the various parameters of each command. Parentheses are not usually required; however, they are shown in the syntax diagrams and examples in the interest of precision.

The following example shows a command coded with parentheses and the equivalent command coded without parenthesis (a more readable format):

```
ADMIN DB LIST S(* FILE(FSTREAM.**) BLOCKS(NOT GT(100)))
ADMIN DB LIST S * FILE FSTREAM.** BLOCKS NOT GT 100
```

Parentheses may sometimes be required to avoid ambiguity. An ambiguous command may result if a keyword value happens to be the same as a keyword. For example, if a job name (used to create an ExHPDM stream file) happened to be called EXPIRED, then the following command would be ambiguous without parentheses:

```
ADMIN DB LIST C DSN ** JOBNAME EXPIRED
```

The minimum parentheses required:

```
ADMIN DB LIST C DSN ** JOBNAME(EXPIRED)
```

The following syntax diagram shows the command formats accepted by SOVADMN. The details of these commands are described in following sections.

Syntax

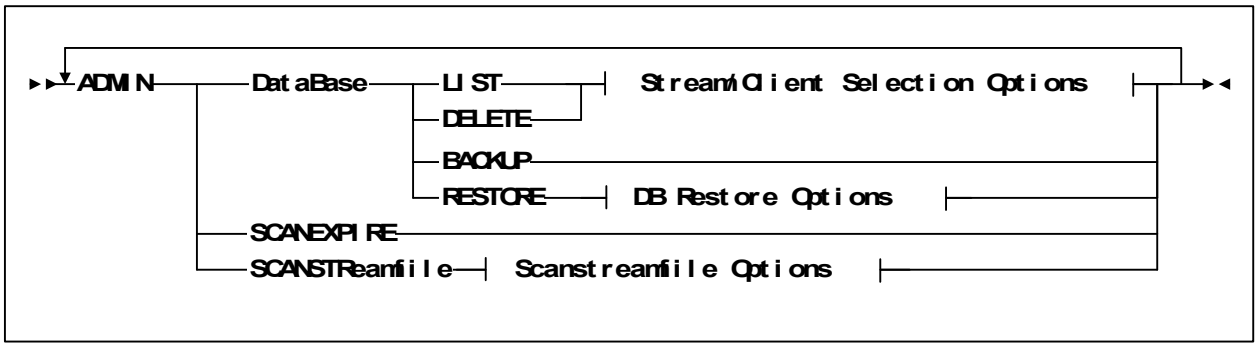


Figure 36. Admin Format Diagram

Database Administration Commands

The ExHPDM database is administered via the following ADMIN commands:

- ADMIN DataBase LIST
- ADMIN DataBase UPDATE
- ADMIN DataBase DELETE
- ADMIN DataBase BACKUP
- ADMIN DataBase RESTORE.

The first three of these, LIST, UPDATE, and DELETE, are discussed together, since they select database records in an identical manner. BACKUP and RESTORE are used respectively to snapshot and to restore the ExHPDM database state. The BACKUP command also performs some necessary reorganization of the database.

DATABASE LIST, UPDATE, and DELETE Commands

LIST, UPDATE, and DELETE commands accept identical parameters, ignoring those that are not applicable. Stream files only, client files only, or both stream files and client files may be selected, as shown in the following syntax diagrams.

Note: The keyword 'DataBase' may be shortened to DB. Other keywords with mixed capitalization may be similarly abbreviated.

The following syntax diagram shows the DataBase LIST, UPDATE, and DELETE format. The Stream Selection Options are described in “ADMIN DATABASE STREAM selection,” Figure 38. on page 221. The Client Selection Options are described by “ADMIN DATABASE CLIENT selection options” on page 221.

Syntax

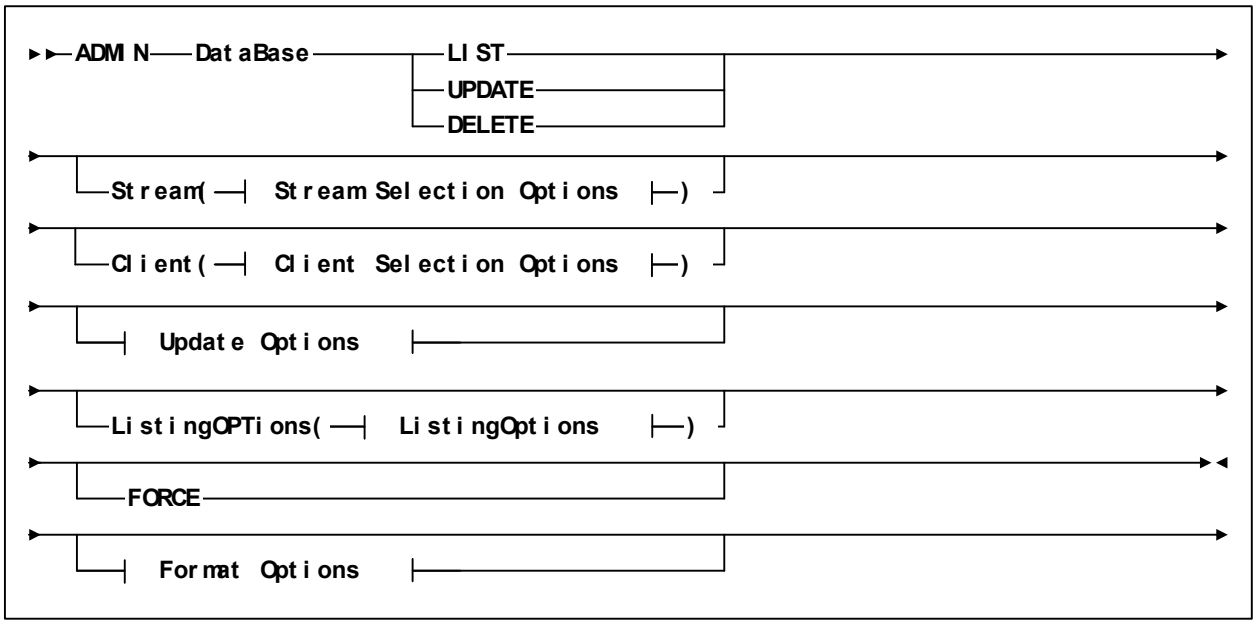


Figure 37. ADMIN DATABASE LIST, UPDATE, and DELETE

LIST

The ADMIN DataBase LIST command has several versions. If only STREAM is specified and not CLIENT, complete stream files are listed. If only CLIENT is specified and not STREAM, only the specified client files are listed. If both CLIENT and STREAM are specified, only the client files in those stream files are listed. If neither CLIENT or STREAM is specified, the default lists the most recent generation of all client files.

UPDATE

The purpose of the UPDATE command is to alter the expiration characteristics of a selected set of stream and/or client files. Where the naming of streams is altered in the startup parameter file and existing streams must be renamed to conform to the change, the UPDATE command allows the user to change the stream name. Where administrative needs or the need to tag particular client files require it, the UPDATE command allows the user to change the client owner name.

When REFORMAT is specified, and no other parameter are specified, all streams are updated.

Listings resulting from the UPDATE command are in the same format as the corresponding LIST command results, except that the UPDATE command results may show client status as “updated.” The list resulting from the UPDATE command shows the results after performing the UPDATE. Where the line is marked “UPDATE FAILED,” the listing shows the unaltered object.

Hint: It is acceptable to place the new keywords in a LIST or DELETE command. This makes it easy to switch between LIST and UPDATE by altering a single

keyword. The new keyword values are ignored for LIST or DELETE; the action is performed only for an UPDATE.

DELETE

The ADMIN DataBase DELETE command deletes everything that the equivalent DataBase LIST command would list.

DELETE FORCE

The DELETE command without the FORCE keyword will not delete steam file records that have apparently not completed. This prevents the creation of orphan clients. The listing obtained by means of the DELETE command indicates stream file records that were not deleted because they were apparently "in progress." "In progress" status is also reported as a result of an ordinary DATABASE LIST command.

The FORCE keyword prevents the check described in the preceding paragraph. FORCE applies only to DELETE commands. It is silently accepted in LIST and UPDATE commands but has no effect.

Caution: FORCE should be used with caution, and only when deleting an uncompleted stream file record is necessary, for example, where the ExHPDM address is cancelled during a stream file write.

Notes:

1. If ADMIN DataBase LIST is specified, all parameters in this command are optional. However, if ADMIN DataBase DELETE is specified, then a STREAM or CLIENT parameter must also be entered.
2. The ADMIN DataBase DELETE command logically removes database entries from the ExHPDM database. The parameters for DELETE are identical to those for LIST. Therefore, the LIST parameter can be used to verify that the database elements to be deleted are the desired set of client and/or stream files. Once verified, the LIST parameter can be changed to the DELETE parameter and the job resubmitted. However, if no parameters are specified with the LIST parameter (list all clients and streams), you cannot change the LIST parameter to the DELETE parameter without adding other parameters. You receive an error message if this is attempted.
3. CLIENT and STREAM parameter lists can be combined. If both are used, the stream parameters are used to extract a subset of the selected client files.
4. When client datasets are deleted using the ADMIN DataBase DELETE command they are only logically deleted from the ExHPDM database. Additional database records are created at the delete time to indicate that the client data sets are deleted. Therefore, the action of deleting client data sets increases the number of records in the database and could trigger the MONitor thresholds if they are reached and result in a console warning message. The actual client data set records are not physically deleted from the database until the subsequent ADMIN DataBase BACKUP command is issued.

5. With the FORCE keyword, if a stream file record is deleted while a client is being written, ExHPDM checks whether the parent stream file record still exists. If the parent record does not exist, the client will not write its own record to the database.

6. The UPDATE command may be issued at any time, even when other activity is updating the database. To protect the integrity of the database, where two or more update operations select the same record at the same time, only one will be successful. When an update fails, the admin job receives a Return Code 4. This is a normal occurrence and does not indicate that the database has been damaged. When an update fails, retrieve the listing to determine whether to edit the update job and resubmit it.

   The most likely cause of a failed update is two update commands running at the same time selecting the same set of records. Selection of an object in the process of being built up by a currently active write stream can also cause an update to fail.

   All generations of a particular client file are stored in one record. Thus update jobs that update different generations of the same client may conflict.

   Where the user requires 100% update reliability, ensure that only one update job is running at a time, that there are no active write streams, and that no DELETE/SCANEXPIRE jobs are running.

When all clients are deleted from a stream file, that stream file is automatically deleted. This condition is checked after any client data set is deleted.

When a stream file is deleted either directly, by not specifying a client, or indirectly, by deleting all the clients stored on that stream file, the volume serial numbers that comprised that stream file are available to be returned to the scratch pool. The volumes that are freed in this way are always listed in the returned output from the ADMIN command.

Additionally, tape volumes are noted for scratching on the MVS MCS console. For example:

```
19.09.13  STC00831  SOV06958I Volume 542628 is now scratch.
19.09.13  STC00831  SOV06958I Volume 542630 is now scratch.
19.09.13  STC00831  SOV06958I Volume 542632 is now scratch.
```

If specified, ExHPDM notifies the TMS via the TMS SCRatchVOLumeRouTiNe for each volume that is scratched. This allows the TMS to reuse these volumes. Refer to the “TMS Keyword” on page 155 for details.

Selecting Streams or Clients

The LIST, UPDATE, and DELETE commands can be made to operate on either stream files or client files, depending on user preference. Stream files contain one or more client files); client DSNs may have several generations.

In most cases, it is preferable to deal with client DSNs only, leaving the administration of stream files up to ExHPDM itself. This is convenient, since it requires no further concepts to be grasped beyond what is already familiar: cataloging (like MVS catalogs) and generations (like MVS GDGs).

For this reason, the 'client' forms of the LIST and DELETE commands are described first.

Working with Client Files

The forms of commands to be used are “ADMIN DataBase LIST”, “ADMIN DataBase UPDATE”, or “ADMIN DataBase DELETE”, followed by CLIENT with a parenthesized set of parameters, as shown in the following syntax diagram. “Client Selection Options” is part of the ADMIN DataBase command.

The ADMIN Database LIST command has several variations. If you specify STREAM only, ExHPDM lists complete stream files. If you specify CLIENT only, ExHPDM lists complete client files. If you specify both STREAM and CLIENT, only the client files in the selected stream files are listed. If neither STREAM nor CLIENT is specified, the default lists the most recent generation of all client files.

Syntax

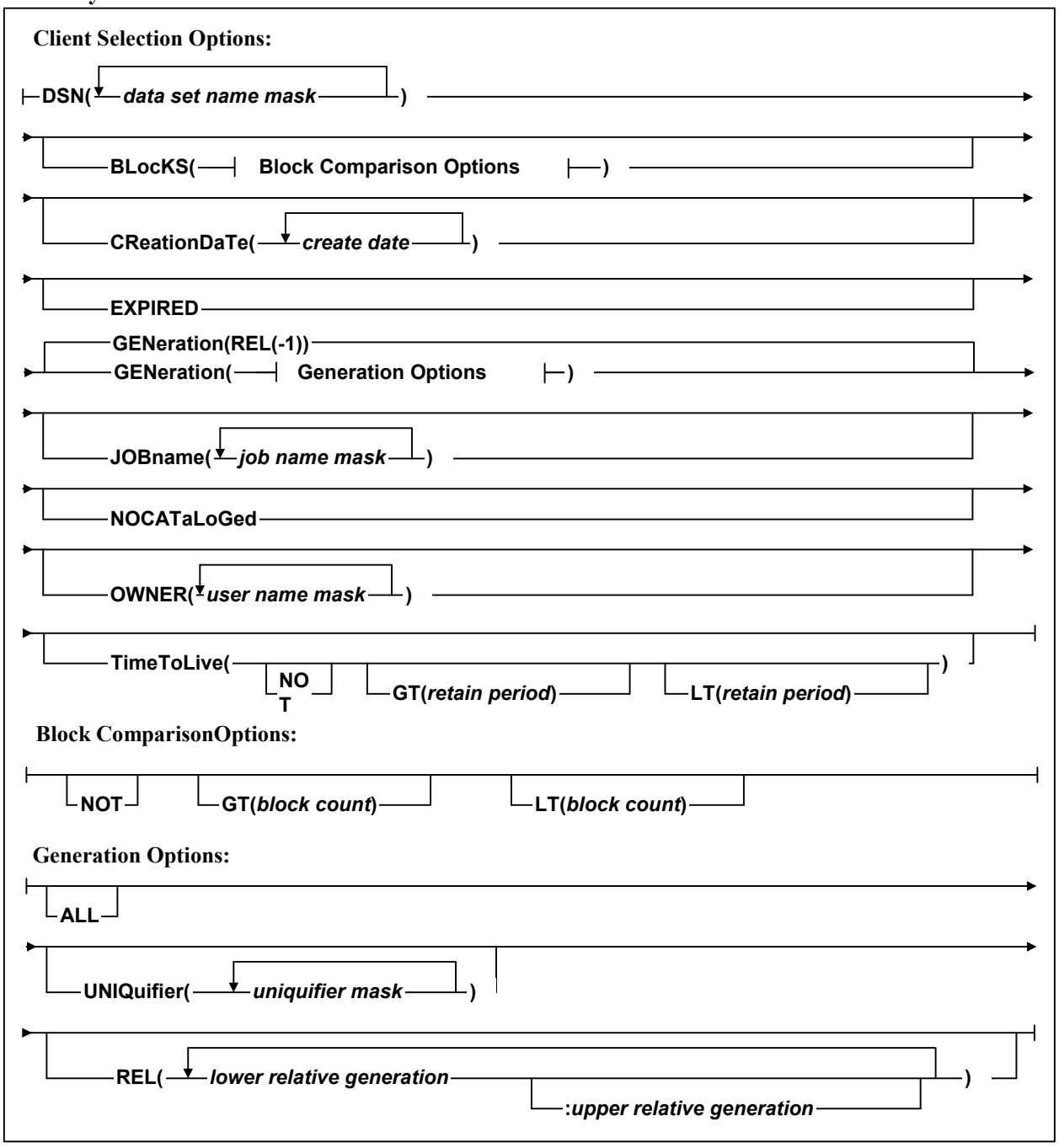


Figure 38. ADMIN DATABASE CLIENT selection options

The following example lists the most recent generation of all clients whose first qualifier is 'PROD01':

```
ADMIN DB LIST CLIENT(DSN(PROD01.**))
```

To extend the above command to include all generations (instead of just the latest), use:

```
ADMIN DB LIST C( DSN(PROD01.**) GEN(ALL) )
```

Note: DSN is a mandatory parameter; that is, ADMIN DB LIST C(GEN(ALL)) is not allowed. If it is indeed required to list clients regardless of their DSN, use the following form:

```
ADMIN DB LIST C(DSN(**)).
```

This uses wildcards to specify all DSNs. Combining this with the GEN(ALL) parameter creates a query which lists the entire database from the client file point of view:
ADMIN DB LIST C(DSN(**) GEN(ALL))
This is an expensive command, since it needs to read practically the entire database into virtual memory.

All other subparameters of 'Client' are optional. If subparameters are omitted, the usual default is to perform no subsetting based on that parameter. The exception to this is GEN, in which case the default is to select only the latest generation rather than every generation.

Client Selection Parameters

DSN

data set name mask specifies a client data set name which may contain masking characters. One or more *data set name mask* values may be specified, separated by spaces.

Masking characters as defined in Table 4. on page 138 may be used in the *data set name mask*. For example:

- PROD01.* is any two level data set name with a high level qualifier of PROD01
- PROD01.SYS%%%.VT*.** would match any three or more level data set name with a high level qualifier of PROD01, a 2nd level qualifier of 6 characters starting with 'SYS', and a 3rd level qualifier starting with 'VT'
- ** matches to all client data set names.

Note: DSN is the only mandatory parameter for Client. To select all client data set names, specify DSN(**).

BLocKS

Allows selection of clients based on their size (number of blocks). Note that this size does not necessarily relate to the amount of physical medium used by the client, since different client data compress differently. However, this is the best measurement currently available.

```
ADMIN DB LIST C(DSN(NETNANNY.LOG.**) BLKS(GT(500)))
ADMIN DB LIST C(DSN(BACKUP.**) BLKS(GT(100) LT(20000)))
ADMIN DB LIST C DSN BACKUP.** BLKS NOT GT 100 LT 20000
```

The first line in the example above finds clients with more than 500 blocks; the second finds clients with between 100 and 20000 blocks inclusive; the third finds clients with less than 100 or greater than 20000 blocks. The optional parentheses have been left off the third line to show that parentheses are not required.

CReationDaTe

Allows selection by creation date of the client file. The parameter value is a list of dates or date/times. The meaning of the list depends on whether there is an odd or even number of entries in the list.

For each pair of entries, the client is selected if its creation date is between the times represented by the pair of entries *create date 1* and *create date 2*. For the last 'odd' entry (if there is one), the client is selected if it was created after the time represented by that entry up to the present time. The following shows examples of using the creation date. A more complete definition of the date and time values that may be specified follows these examples.

```
ADMIN DB LIST C( DSN(**) CRDT(15JUL2000) )
```

selects clients created after midnight, 15 July, 2000. The date must be specified as two day digits, three letter month abbreviation, then 4 digit year.

```
ADMIN DB LIST C( DSN(**) CRDT(01JAN1999 01JUL1999) )
```

selects clients created between midnights of 1 January and 1 July, 1999. Note that this does not include the day of 1st July itself.

```
ADMIN DB LIST C( DSN(**) CRDT('02AUG1999 13.30.00') )
```

selects clients created after 1:30pm on 2nd August, 1999.

```
ADMIN DB LIST C( DSN(**) CRDT(-2) )
```

selects clients created after midnight two days ago.

```
ADMIN DB LIST C( DSN(**) CRDT(01JAN1998 YESTERDAY) )
```

selects clients created after midnight, 1 January 1998 up to, but not including, yesterday.

```
ADMIN DB LIST C( DSN(**) CRDT(01JAN1998 'YESTERDAY 12am') )
```

selects clients created after midnight, 1 January 1998 up to 12 noon yesterday.

The create date values may be specified using a date and a time format. The date and time values may be combined to form a required creation date value. Any number of spaces may be used to separate the time from the date. If spaces are used, then the date and time must be enclosed in quotes (‘’). If a time is specified without a date, the

default date is today. If a date is specified without a time, the default time is midnight. The following describes the acceptable forms for date and time.

Date Formats

The following field variables are used to describe valid dates formats. Where the explicit specification is given, this indicates that the value may be specified anywhere within the date:

- *yyyy* or *yyyy*Y, as the explicit specification, indicates a two- or four-digit year. If a two-digit year is specified then the century switch is based on the year. Years 40 to 99 are assumed to be in the 1900's, and years 00 to 39 are assumed to be in the 2000s.
- *jjj* or *jjj*J, as the explicit specification, indicates a day within a year.
- *dd* or *dd*D, as the explicit specification, indicates a day within a month.
- *mm* or *mm*MONTH, as the explicit specification, indicates the month number from 1 to 12.
- *mon* indicates a month name as follows. Capitalization indicates the required part of the name that may be specified. JANuary, FEBruary, MARch, APRil, MAY, JUNe, JULy, AUGust, SEPtember, OCTober, NOVember, DECember.

Where the explicit specification of the above variables is used then the dates may be specified in any format. For example:

1MONTH2D1999Y

is the 2nd day of January 1999.

The following date formats may be specified where the explicit forms of the date variables is not used:

yyyymmdd

International date format. For example:

20000102
is the 2nd day of January 2000.

19991220
is the 20th day of December 1999.

991220
is the 20th day of December 1999 using a two-digit date.

yyyyjjj

Julian date format. For example:

2000003
is the 3rd day of January 2000.

99365

is the last day of 1999.

mm/dd/yyyy
US date format. For example:

03/27/1964
is the 27th day of March 1964.

02/04/02
is the 4th day of February 2002.

dd/mon/yyyy

ddmonyyyy

mon/dd/yyyy
Date format using an explicit month name. For example:

SEP/17/1989
is the 17th day of September 1989.

09FEBRUARY94
is the 9th day of February 1994.

TODAY
Shortcut to indicate today's date.

YESTERDAY
Shortcut to indicate yesterday's date.

TOMORROW
Shortcut to indicate tomorrow's date.

+nn
Relative date format indicating the number of days *nn* in the future. For example:

+2
is two days from today's date.

+234
is 234 days from today's date.

-nn
Relative date format indicating the number of days *nn* in the past.

-1
is yesterday's date.

-29
is 29 days ago.

0
Synonym for TODAY.

Time Formats

The following field variables are used to describes valid time formats:

- *hh* or *hh*H, as the explicit specification, indicates hour.
- *mm* or *mm*M, as the explicit specification, indicates minutes.
- *ss* or *ss*S, as the explicit specification, indicates seconds.

Where the explicit specification of the above variables is used then the times may be specified in any format. For example:

20S10H
is the 20 seconds past 10 in the morning.

The following time formats may be specified when the explicit form is not used. A The usage of AM and T is optional. If an *hh* (hour) value of between 1 and 12 is specified, then AM is assumed, otherwise a value of T is assumed:

hhAM

hh.mmAM

hh.mm.ssAM
Morning 12-hour time. For example,

12am
is noon.

9.20
is 20 minutes past 9 in the morning.

10.30.02am
is 2 seconds after 10:30 in the morning.

hhPM

hh.mmPM

hh.mm.ssPM
Afternoon 12-hour time. For example,

12pm
is midnight.

hhT

hh.mmT

hh.mm.ssT
Military (24-hour) time. For example,

22.01T

13.01

Combined Date and Time Formats

Any combination of the date and time formats may be specified as described in “Date Formats” on page 224 and “Time Formats” on page 226. If only a time is specified, then the date defaults to TODAY. If only a date is specified, then the time defaults to midnight on that date. The two specifications may be separated by any number of spaces. If spaces are used then the date and time must be enclosed in quotes (‘’). For example:

‘Today 10am’
10.00am, today.

‘13.00.00T Tomorrow’
1.00pm. tomorrow.

‘-2 10pm’
10.00pm, 2 days ago.

In addition the following combined dates and times may be specified:

NOW
Current date/time.

yyyymmddhhmmss
International date/time format. For example,

19970203130000
is 1.00pm on the 03 February 1997.

yyyydddhhmmss
Julian date/time format. For example,

94123130000
is 1.00 pm on the 03 May 1994 (i.e., 1994 day 123).

Note: Note that a date specified as 01OCT1997 represents the start of that date at midnight. Therefore, a range of dates, 05AUG1997 10AUG1997 selects clients that were created between midnight 5 August and midnight 10 August. This does not include clients created on the day of 10 August. To be specific about the creation date, you can code explicit times on the date ranges. For example, ‘05AUG1997 00h’ ‘10AUG1997 23h59m59s’.

EXPIRED

Allows selection of client files which have expired. This allows ExHPDM to automatically manage client file generations via 'management' options. Clients may be set to expire after a certain time period from their creation, when a particular number of more recent generations exist, or when a client file is uncataloged.

ADMIN DB DELETE C(DSN(**) EXPIRED)

If this command was submitted regularly, it would allow ExHPDM to delete all expired client files from its database. In time, this would allow volumes which contained expired clients to be recycled (scratched). This format of the command is equivalent to using ADMIN "SCANEXPIRE" on page 265.

Note: Note that, unlike all other subparameters, the default GEN subparameter is ALL if EXPIRED is specified.

GENeration

specifies which ExHPDM generation(s) of the client data set(s) is required. These can be selected by either a range of relative generation numbers or a uniquifier value. By default, only the latest generations are selected unless the EXPIRED parameter is used, in which case it is more sensible to default to all generations.

Generations can be selected by relative generation number (counting either from the most recent or the most ancient), or by specific uniquifier.

Note: The **GENeration** parameter is not related to MVS Generation Data Groups (GDGs). **GENeration** is an ExHPDM term indicating successive client data sets, with the same name, saved in a stream file.

REL

specifies a range of relative generations that may be selected. The relative generations may be selected by counting from the most recent (negative value) or the most ancient (positive value) generation. If a range of relative generations is required then the range is separated by a colon.

If there are *n* generations of a client data set then REL(–1) is the most recent generation, REL(–2) is the second most recent, and REL(–*n*) is the most ancient generation.

This may also be expressed as REL(1) being the most ancient generation, REL(2) being the second most ancient, and REL(*n*) being the most recent generation.

Note: Note that when specifying the REL parameter, numbers outside the allowable range are allowed but ignored. For example, REL(1:9999) is allowed even if there are fewer than 9999 generations.

Also, '0' is allowed as the second part of a range, but not as the first part. For example, REL(–5:0) is equivalent to REL(–5:–5) or just REL(–5). The two parts of a range can be entered in ascending or descending order. That is, REL(8:1) is equivalent to REL(1:8). Ranges with mixed positive and negative specifications are not allowed. For example, REL(–1:1) is an error.

REL(0) is not a valid relative generation.

ALL

specifies all generations.

UNIQuifier

uniquifier mask specifies specific client instance(s). ExHPDM assigns a unique 16 digit hexadecimal when a client requests connection to ExHPDM. This value defines a unique instance of a client data set within the ExHPDM database.

uniquifier mask may be specified as an ExHPDM maskable value as described in Table 4. on page 138.

Select clients starting with HTMLLNKS with either the latest or second most recent generation:

```
ADMIN DB LIST C(DSN(HTMLLNKS.**) GEN(REL(–1:–2)))
```

As above, except selects only the second oldest through to fourth oldest generations:

```
ADMIN DB LIST C(DSN(HTMLLNKS.**) GEN(REL(2:4)))
```

Select with specific uniquifier. Wildcards can be used:

```
ADMIN DB LIST C(DSN(HTMLLNKS.**) GEN(UNIQ(AF01E577D*)))
```

```
ADMIN DB LIST C(DSN(HTMLLNKS.**) GEN(ALL))
```

If both GENERATION and CREATIONDATE parameters are specified, the resulting client instances match both criteria.

Any combination of REL, ALL and UNIQuifier can be specified, and the result is the OR of all options.

If you specify the GENERATION parameter without any of its subparameters, GENERATION is ignored.

GENeneration(REL(–1)), the most recent client generations, is the default.

JOBname

Allows selection of clients created by particular job name(s) as specified on *job name mask*. The parameter value may contain one or more strings, optionally with wildcards.

```
ADMIN DB LIST C( DSN(CICS.DW.**) JOB(CICSBACK CICSBK*) )
```

lists the latest generation of clients DSNs starting with CICS.DW, and created by jobs called CICSBACK or a job starting with CICSBK.

job name mask may be specified as an ExHPDM maskable value as described in Table 4. on page 138.

NOCATaLoGed

Allows selection of clients which are not currently cataloged in the MVS catalog. The purpose of this option is to simplify administration of client files by allowing use of MVS catalog operations. Normally, when a client

file is created by ExHPDM, an entry for it is cataloged with a unique volume serial identifier which indicates that it is managed by ExHPDM. The user may then delete this file by uncataloging it. ExHPDM may then use this information to automatically delete its database entries for the client file.

```
ADMIN DB DELETE C(DSN(**) NOCATLG GEN(–2:–99))
```

If this command was submitted regularly, it would allow ExHPDM to be kept up to date with respect to client files which were uncataloged (i.e., deleted), except that the latest generation is kept regardless of its catalog status.

OWNER

Allows selection of clients created by particular user IDs. Usage is very similar to JOBname.

user name mask may be specified as an ExHPDM maskable value as described in Table 4. on page 138.

TimeToLive

Allows selection of clients based on their expected time until they expire. The logic of the GT, LT and NOT keywords is the same as for BLocKS. The time to live (*retain period*) for a client is negative if it has already expired (by date), zero if it has expired because of generation count or uncataloging. If the client does not expire, its time to live is deemed infinite.

```
ADMIN DB LIST C(DSN(**) GEN(ALL) TTL(LT(7D)))
```

This finds clients due to expire within one week (7 days). Note that the 'D' unit is required, since the default time unit for all ExHPDM commands and parameters is 'seconds'. *retain period* is specified as a time interval as described by “Specifying a Time Interval or Period” on page 161.

Working with Stream Files

The form of command to be used consists of “ADMIN DataBase LIST”, “ADMIN DataBase UPDATE” or “ADMIN DataBase DELETE” followed by STREAM with a parenthesized set of parameters, as shown in the following syntax diagram.

“Stream Selection Options” is part of the ADMIN DataBase command.

The ADMIN Database LIST command has several variations. If you specify STREAM only, ExHPDM lists complete stream files. If you specify CLIENT only, ExHPDM lists complete client files. If you specify both STREAM and CLIENT, ExHPDM lists only the client files in the selected stream files. If you specify neither STREAM nor CLIENT, the default lists the most recent generation of all client files.

Warning: Using the CLIENT parameter is not advised, because it produces listing by client only.

All other subparameters of STREAM are optional. If a parameter is omitted, the default is to perform no subsetting based on that parameter.

Syntax

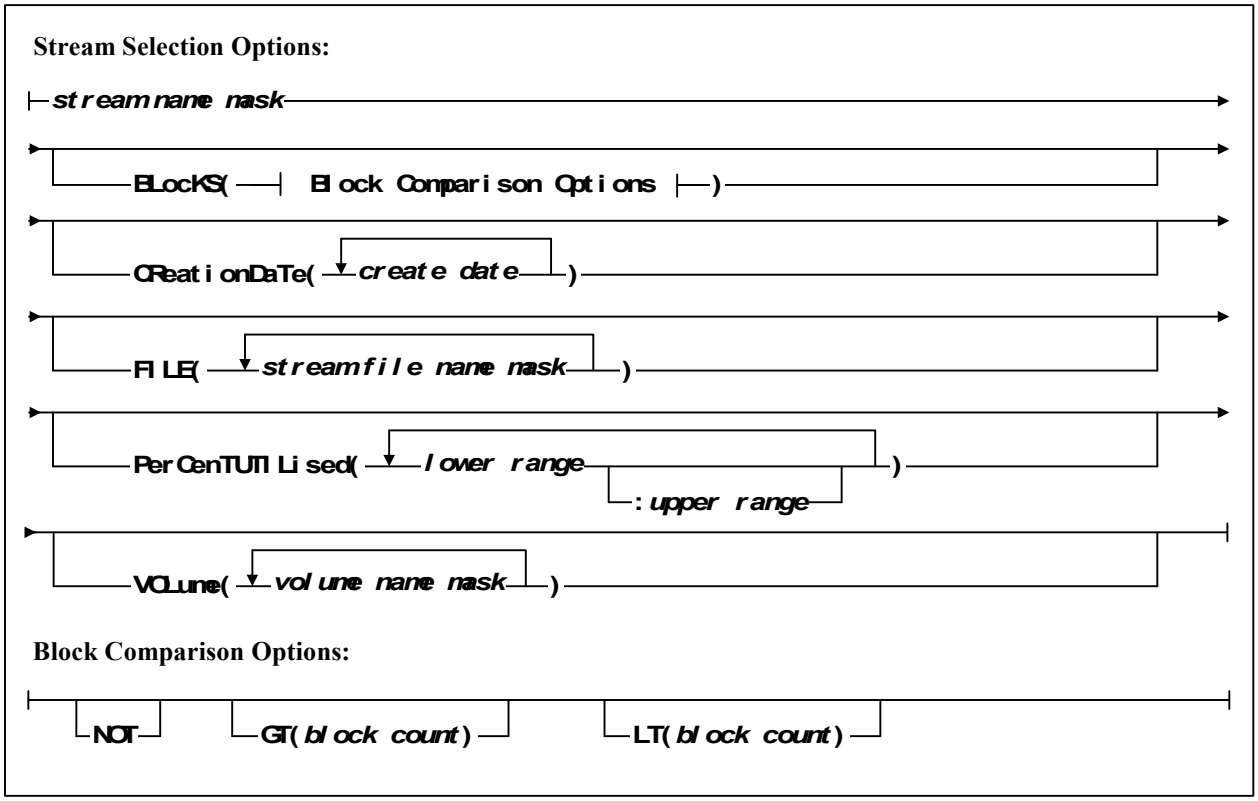


Figure 39. ADMIN DATABASE STREAM selection

Stream FileSelection Parameters

stream name mask

This positional parameter specifies the stream name(s) to be selected. *stream name mask* may be specified as an ExHPDM maskable value as described in Table 4. on page 138.

ADMIN DB LIST S(FAXSTRM*)

selects streams whose name (as specified in the ExHPDM parameter file) starts with FAXSTRM.

BLocKS

operate like the Client subparameter of the same name, except that this parameter applies to the stream file as a whole.

CReationDaTe

operates like the Client subparameter of the same name, except that this parameter applies to the stream file as a whole.

FILE

This parameter specifies the DSN of the stream file(s) to select. *stream file name mask* may be specified as an ExHPDM maskable value as described in Table 4. on page 138, and may be specified as a list of values, separated by spaces.

```
ADMIN DB LIST S(FILE(EXHPDM.FAXSTRM.**))
```

selects stream files whose DSN starts with EXHPDM.FAXSTRM.

PerCenTUTILised

This parameter specifies the percentage utilization of the stream file. The utilization is defined as the number of blocks in all accessible clients divided by the number of blocks written to the stream file when it was created. A client is accessible if it has not been deleted and it was written successfully in the first place. This form of the command is used to determine which stream files are almost fully utilized. The few remaining clients in such stream files are regenerated onto a new stream file; then the original stream file is deleted, thus freeing its volumes.

```
ADMIN DB LIST S(PCTUTIL(20))
ADMIN DB LIST S(PCTUTIL(80:100))
```

The first example selects stream files which are less than 20 percent utilized, and the second example selects those between 80 and 100 percent utilized.

Note: Where the percent utilization of a stream file is less than 1 percent it is rounded up to 1 percent. For example, a utilization of 0.1 percent is rounded up to 1 percent. Only a completely empty stream file is reported as 0 percent.

VOLume

This parameter specifies one or more volumes. It selects stream files that contain those volumes. This is most commonly used to correct the ExHPDM database in the case that one or more volume serials were scratched without ExHPDM being aware. *volume name mask* may be specified as an ExHPDM maskable value as described in Table 4. on page 138, and may be specified as a list of values, separated by spaces.

```
ADMIN DB DELETE S(VOL(HPDM01 HPDM05))

lists and deletes stream files that contain either or both of the
volumes HPDM01 or HPDM05.
```

INCOMPLETE

This parameter selects only apparently incomplete stream files. The default is to select both complete and incomplete stream files.

ADMIN Database LIST, UPDATE, and DELETE Examples

Example 1

The following example lists all the data sets managed by ExHPDM with data set names that start with “OWPL.IMAGE...”. Only the most recent generations are listed.

```
ADMIN DATABASE LIST C(DSN(OWPL.IMAGE.**))
```

Example 2

The following example lists all the ExHPDM stream tape volumes that have more than 80 percent disused space in them. Disused space means that a previous DELETE has removed some of the client files that were previously on a stream file. The percentage figure is calculated as the current number of blocks of all nondeleted clients divided by the number of blocks originally written to the stream file. When specifying the PERCENTUTILised parameter, if only a single figure is given, as in the following example, any utilization less than or equal to the figure is selected. If a range is given, as in the other example, the utilization must fall within that inclusive range.

```
ADMIN DATABASE LIST STREAM(PERCENTUTIL(20))
ADMIN DATABASE LIST STREAM(PERCENTUTIL(80:100))
```

Example 3

Delete the most recent 3 generations of OWSH data sets.

```
ADMIN DATABASE DELETE CLIENT(DSN(OWSH.**) GEN(REL(-1:-3)))
```

Example 4

This example shows how, for clients stored in TIMSTRM streams only, delete the oldest generation of OWSH data sets or any OWSH data set with a uniquifier starting with AF2365BB.

```
ADMIN DATABASE DELETE
STREAM(TIMSTRM)
CLIENT(
DSN(OWSH.**)
GEN(REL(1)UNIQ(AF2365BB*))
```

Example 5

This example shows how to delete all client data sets owned by OWWT and written by a job called OWWTDSSU.

```
ADMIN DATABASE DELETE CLIENT( DSN(**) GEN(ALL) OWNER(OWWT)
JOB(OWWTDSSU))
```

Example 6

The following example deletes all uncataloged client data sets. This function is useful for purging old or stale MVS generation data group (GDG) entries.

```
ADMIN DATABASE DELETE CLIENT (DSN(**) NOCATLG)
```

ADMIN DataBase UPDATE command

Syntax

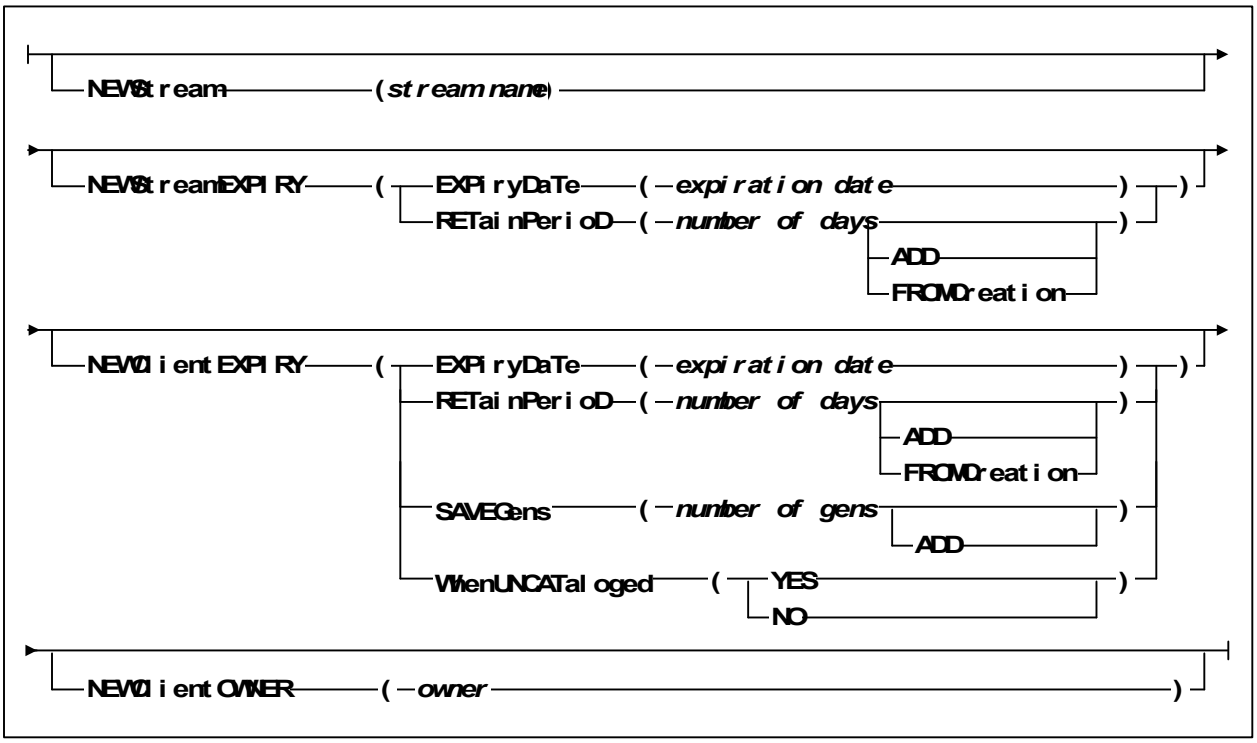


Figure 40. ADMIN DATABASE UPDATE Options

UPDATE Parameters

NEWStream

Rename the stream to the specified stream name.

NEWStreamEXPIRY
Change the stream file expiration characteristics.

EXPiry Date
The new expiry date in yyyy/ddd format.

RETainPerioD
Change the retention period. If ADD is specified, the number of days is added to the current expiry. If FROMCreation is specified, the number of days is added to the original creation date. The default is to add the number of days to the present date

NEWClient EXPIRY
Changes client file expiration characteristics.

SAVEGens
Change the number of saved generations. If ADD is specified, the value is added to the current number of saved generations.

WhenUNCATaloged
Change "whenuncatalogued" expiry.

NEWClientOWNER
Change the owner userid.

ExHPDM has three variations of the command for changing the retain period for a STREAM or CLIENT. If neither ADD nor FROMCreation is specified, the default adds the given number of days to the time at which the update job is run. If you specify ADD, ExHPDM adds the number of days to the date at which the object would have expired. If you specify FROMCreation, ExHPDM adds the number of days to the original creation date of the object.

Notes:

1. It is not possible to specify both ADD and FROMCreation.
2. The number of days cannot be expressed as a negative; that is, the ADD parameter can only retard an expiration date, not advance it.

Examples

Example 1

To rename streams that were originally created as TEST streams to PROD:

```
ADMIN DB UPDATE STREAM(TEST) NEWSTREAM(PROD)
```

Example 2

To change the expiry of all generations of PROD.**.BACKUP to 90 days from the date the update job is run:

```
ADMIN DB UPDATE CLIENT(DSN(PROD.**.BACKUP) GEN(ALL))
NEWCEXPIRY( EXPDT(9999/355) SAVEGENS(0) WUNCAT(NO) )
```

Note: A maximum of three generations will be kept. If there are currently more than three generations of such client files the fourth and any older generations will be implicitly expired by this command.

Example 3

To make the latest generation only of PROD.VOL001 ARCHIVE permanent:

```
ADMIN DB UPDATE CLIENT (DSN(PROD.VOL001.ARCHIVE))
NEWCEXPIRY( EXPDT(999/355) SAVEGENS(0) WUNCAT(NO) )
```

Example 4

To ADD one saved generation to all PROD.VOL*.ARCHIVE client files with generation numbers between -1 and -4 inclusive:

```
ADMIN DB UPDATE CLIENT(DSN(PROD.VOL*.ARCHIVE) GEN(REL(-1:-4)))
NEWCEXPIRY( SAVEGENS(1 ADD) )
```

Hint: For example, if one of these client files had a SAVEGENS number of 10, it would be increased to 11.

ADMIN DATABASE Listing Options

SYNTAX

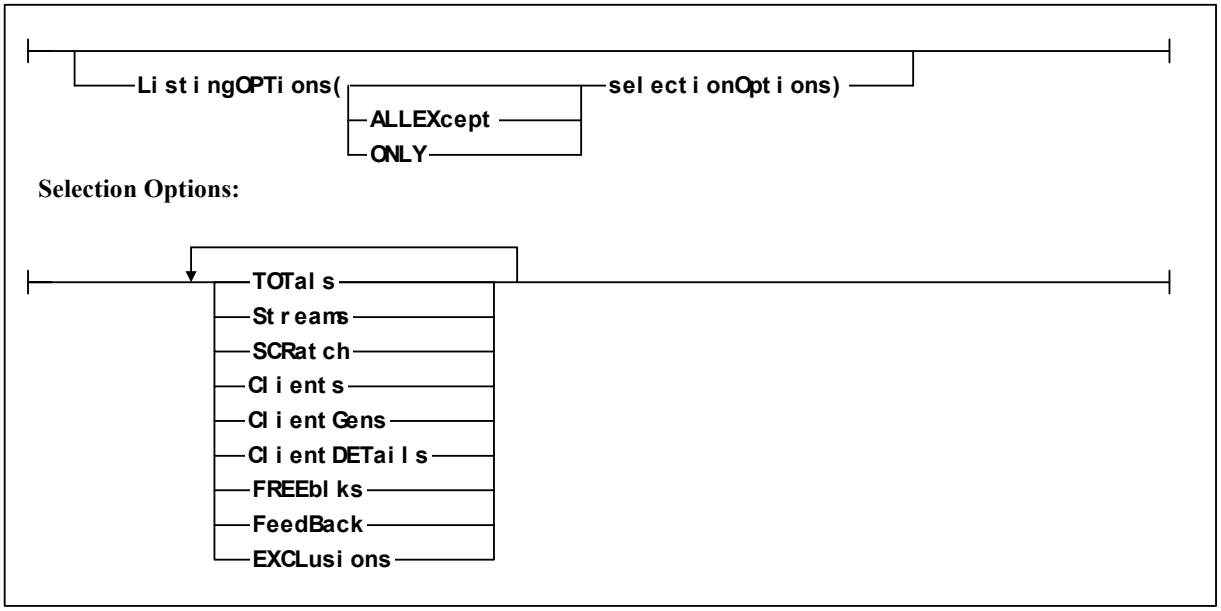


Figure 41. ADMIN DATABASE Listing Options

Listing Options

ListingOPTions

Allows the user to specify, at a coarse level, the tables and informational items to be included or excluded. If ListingOPTions are not specified, the default is to list all available reports.

ALLEXcept

All reports will be listed except for those specified by the following selection options:

ONLY

List only the reports specified by the following options:

TOTals

Show summaries.

Streams

List streams.

SCRatch

For DELETE, list scratched volumes.

Clients

List clients.

Client Gens

List client generations.

Client DETails

List details of specific clients.

FREEblks

List blocks freed by DELETE processing.

FeedBack

Show explanation of LIST / DELETE command.

EXCLusions

Show excluded stream reasons list.

Examples

Example 1

The following parameter specifies that only the stream and first of the client tables be shown:

```
ADMIN DB LIST C(DSN **) LOPT(ONLY S C)
```

Example 2

The following parameter eliminates the initial command feedback (that is, explanation) and the list of reasons for excluding stream files from the list:

```
ADMIN DB LIST C(DSN PROD.**) LOPT(ALLEXCEPT FB EXCL)
```

Output Listing Format

Each LIST or DELETE command produces output which is placed in the SYSPRINT DD. LIST and DELETE produce similar output. Differences are explained below.

The listing output is composed of three major parts: the header, the stream file section and the client file section.

Header Section

The first part of the header is an explanation of the command that was used. Inspect this part to determine how ExHPDM interpreted the command. (Since the various parameters sometimes interact, the output may not be what you expected.)

The second part of the header gives statistics for the database as a whole, such as the number of stream files, the total number of known volume serials, and an average computed from these.

ExHPDM lists any volume serials in the database that do not belong to any known stream file. these volume serials are listed. Normally there should be no such volumes. Delete any such volumes from the database using

ADMIN DB DELETE S(VOL(*list_of_volsers***))**

as soon as possible.

ExHPDM V3.0 provides an improved report format over the V2.0 report format. By default, the V3.0 report is produced when keywords REPORT FORMAT(V20) is not specified for a V2.0 report format. The V3.0 report format combines the Stream Table 1 with the Stream Table 2 information to present a more readable and usable report. Additionally, all the pages are numbered in the V3.0 report and contain the date the report was run. The Client Table 1, Client Table 2, and Client Table 3 information are also combined.

Stream Section

The next part of the output listing consists of two tables. Stream Table 1 describes the stream DSN, specifying, among other things, when it was created. Stream Table 2 details the volume serials and expiry information. The contents of these tables are described below:

Stream Table 1

SRef
A number, used only for the current listing output, which identifies a particular stream file. Subsequent tables use this number when referring back to a particular stream file.

Stream
The name of the stream, as provided in the stream definition in the ExHPDM startup parameter file.

Stream file
The full stream file DSN. Note that the last qualifier level is a unique identifier generated by ExHPDM each time it creates a new stream file.

Created
The date on which this stream file was created. DDMONYYYY format.

Devt
The device type of the device that the stream file was written.

Esoteric
The esoteric device name of the device that the stream file was written.

Vols
The total number of volumes that this stream file was written to. This is a number from 1 to 255, with the upper limit being determined by MVS limitations.

Clnts
The total number of client files written to this stream file. This includes clients which did not complete successfully.

Util
The current percentage utilization for the stream file, as computed based on the next two fields.

UsedBlks
The number of blocks currently used in the stream file. This is the total block count for all accessible clients. Accessible clients are those clients which have not been deleted and were written successfully in the first place.

TotlBlks
The total number of blocks initially written to this stream file when it was created. This includes blocks taken up by clients that were subsequently deleted, as well as clients that were not written successfully.

Stream Table 2

SRef
This number refers to the stream file in Stream Table 1.

Stream
The stream name, repeated from Stream Table 1.

Stream file
The stream DSN, repeated from Stream Table 1.

Expiry
The expiry date of this stream file, in the format DDMONYYYY. If the stream does not expire, this field is left blank. If the stream file had its expiry specified as RETPD, this is converted to an expiry date for display in this table.

VOLSERs
On the line(s) following each stream file name, the list of volume serials that comprise this stream file are listed. Up to 12 volume serials are listed on each line.

Client section

Two or three client file tables may follow Stream Table 1 and Stream Table 2.

- Client Table 1 lists the clients that were directly selected by the query.
- Client Table 2 gives more details about those clients, plus other clients which happen to be in the same stream file as the clients that were explicitly selected. The clients which were explicitly selected are marked as such in the 'status' field.
- The third table, Client Table 3, is produced only if the DSN subparameter of Client contained only fully qualified names (i.e., no wildcards) and contains details of interest for a more specific search.

Client Table 1

Client DSN
The data set name of the client file. This is not necessarily unique, since several generations of the same client DSN can exist at the same time.

SRef
A reference to Stream Table 1, indicating the stream file in which this client file belongs.

Created
The creation date of the client file, in DDMONYYYY format. This would normally be the same as the stream file creation date, unless the stream was very long lived.

Owner
The MVS userid of the job that created this client file.

Jobname
The name of the job which created this client file.

Vols
The total number of volumes that this client file covers. This may be less than or equal to the number of volumes in the stream file.

Recfm
The record format of the client file.

Blksz
The block size of the client file. This is not related to the ExHPDM block size (which is always 256K), but to the block size as perceived by the client job.

Lrecl
The logical record length of the client file.

Blocks
The total number of ExHPDM blocks used by this client. This number is used in the computation of the relative amount of space taken up by this client file. This does not necessarily relate to the amount of physical medium consumed, since blocks from different clients may be compressed in the backend with varying degrees of efficiency.

Start
The starting volume number and block identifier for the first block of this client. This is formatted as volno, blkid where volno is a number from 1 to 50 and blkid is a hexadecimal number of the block on this volume.

End
The ending block of the client. Currently, this value is meaningless since the actual ending block value is not stored.

Client Table 2

Client DSN
Repeat of the client DSN from client table 1.

SRef
Repeat of the stream file reference number from client table 1.

Uniquifier
The uniquifier of this client. This is a 16 digit hexadecimal number. It is generated directly from the MVS time of day clock when the client is created, with additional uniquifying bits. If necessary, a particular client file can be accessed using this number, since each client is uniquely identified for all time by this number.

+Gen
The generation number of this client, counting from the most ancient accessible generation. The oldest client is numbered '1', with successively more recent generations being assigned higher numbers. Unlike the uniquifier, this number only refers to a particular client at a particular point in time. If clients with the same DSN are deleted, then the generational numbering is updated to reflect the change.

-Gen
The generation number of this client, counting from the most recent accessible generation. The most recent client is numbered '–1', with successively older

generations being assigned more negative numbers. As for +Gen, this number is associated with a particular client only transiently.

SGen

For client expiration, if the client was specified as expiring when a particular number of more recent generations were created, this number indicates how many newer generations are allowed before expiring this client. If zero, then there is no generational expiry. In addition, if the client is to expire when it is uncataloged, this field contains a 'U' in the first position. In addition, for the format V20 report, if the client is to expire when it is uncataloged, this fields contains a “U” in the first position.

Expires

If the client expiry was specified as a date, this field indicates the date of expiry. If the field is left blank, there is no expiry date.This field may indicate either an expiry specified when the client was created, or an expiry date derived from the stream file expiry. In addition, for the format V30 report, if the client is to expire when it is uncataloged, this field contains the word CATALOG for client datasets under catalog control (WhenUNCATaloged), or the word PERMANENT for those client datasets that never expires.

Status

Indicates the status of this client file. The contents of this field depend on whether this is a LIST or a DELETE operation.

For LIST, the status field contains one of the following:

Selected

The client was selected by the LIST command.

Not selected

The client was not selected by the list command but is listed here because it also resides in a stream file from which at least one other client was selected.

Deleted previously

Similar to 'Not selected', but in addition indicates that this client was deleted by a previous DELETE command. This includes clients that had been terminated before they had completed writing.

NOTAVAIL

A client file is known in the stream file; however, ExHPDM finds no other record of this client file. This indicates an internal inconsistency in the database.

For a DELETE command, the status may be one of the following:

DELETED

The client was explicitly selected for deletion.

KEPT

The client was not selected for deletion, but it exists in a stream file which contains at least one other client file that was selected for deletion.

Deleted previously
Similar to 'Not selected', but in addition indicates that this client was deleted by a previous DELETE command.

NOTAVAIL
A client file is known in the stream file; however, there is no other record of this client file. This indicates an internal inconsistency in the database.

In addition to the above status indicators, the status field can also show the expiry status of the client. If expired, the status field also shows a parenthesized string starting with 'expired:'. The possible expiry reasons are:

stream retpd
Expired because the containing stream file has expired.

savegens
Expired because there are sufficient more recent generations of the same client DSN.

client expdt
Expired because of expiry date of the client file itself.

client uncat
Expired because the client file was uncataloged.

Client Table 3

This table appears only when all client DSNs were specified without wildcards and gives additional information pertaining to those client DSNs.Client Table 3 has a section for each specified client DSN and lists within each section all known generations of the specified client DSN, including, possibly, deleted generations.

Client DSN
This is the client's data set name. It is given once at the start of each subtable for the client. The following fields pertain to the listed client:

+Gen
Repeat of the positive generation number in client table 2. In addition, this number may be zero if no generation can be assigned to this client file.

-Gen
Repeat of the negative generation number in client table 2. In addition, this number may be zero if no generation can be assigned to this client file.

Status
Shows the status of this client generation. May be one of:

<blank>
Normal status, client is available.

Deleted
Client was previously deleted. This includes clients that had been terminated before they had completed writing.

Orphaned
Client has not been deleted, yet the stream file for this client does not exist. This indicates an internal database inconsistency. In this case, the generation number is listed as zero, since the client cannot be accessed by any mechanism.

SRef
Stream file reference number, used to index stream table 1.

Created
Creation date of this client generation, in DDMONYYYY format.

Uniquifier
Client uniquifier, repeated from client table 2.

Stream DSN
Stream file data set name, copied from the entry in stream table 1.

Free blocks
In the case of DELETE commands, if clients were deleted, ExHPDM lists the number of blocks freed up in each stream file. If stream files were deleted, ExHPDM lists the volume serial numbers which were freed. The external data manager (if any) is automatically notified of any scratched volumes.

A stream file may be freed either because it was explicitly selected for deletion, or because all accessible clients were deleted from it. The listing makes clear which case it was that led to the stream file's deletion.

Generating Raw Client or Stream Report Data

The SOVADMN Reporting facility makes raw output available. A DD name or dataset name destination may be specified.

ExHPDM supports three report formats: StreamForClient, ClientForStream, and RePorT. For all formats except RePorT, the data provided is raw; that is, the data is not converted from its existing format.

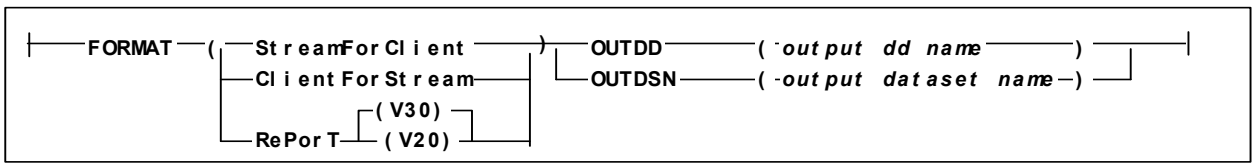


Figure 42. ADMIN DataBase FORMAT Options

FORMAT
Output from the ADMIN DB command is written in one of the following formats.

StreamForClient

Output records present Client data by Stream. See following sections for detail of record formats.

ClientForStream

Output records present Stream data by Client. See following sections for detail of record formats.

RePorT

The output will be produced in a format suitable for printing.

V30

The report will be in the new V3.0 format which includes report titles and page numbering. This is the default format.

V20

The report will be in the old V1.1/V2.0 format.

OUTDD (output DD name)
OUTDSN (output dataset name)

Specifies a DD name, or dataset name that will be used to hold the generated output. This parameter is only used when the format is not RePorT. If an OUTDD is specified, then the value should not be SYSPRINT.

The output file consists of variable length records with a maximum blocksize of 32756.

For StreamForClient and ClientForStream formats, the output is a series of records selected on the basis of other parameters specified on the ADMIN DB command, and presented in the format specified below. Each record contains a record descriptor word preceding the data.

Example 1

The following example lists all the client information for stream STREAM1 in StreamForClient mode. The data is directed to the STRMOUT DDname. A description of the resulting data follows.

```
ADMIN DATABASE LIST STREAM(STREAM1) FORMAT(STREAMFORCLIENT)
OUTDD(STRMOUT)
```

```
SOVSCRDW   DS    F       Length of data (excluding RDW)
SOVCFSID   DS    CL4     Eyecatcher: " SFC"
SOVSNAM    DS    CL8     Stream name
SOVSFNAM   DS    CL44    Stream file name
SOVSSEQN   DS    F       Stream File sequence number
SOVSDATE   DS    2F      Date stream created (TOD format)
SOVSRES1   DS    3F      Reserved
SOVSESOT   DS    CL8     Esoteric name
SOVSUBLK   DS    F       Binary value # of blocks used
SOVSTBLK   DS    F       Binary value Total number of blocks
SOVSEXPT   DS    2F      Stream file Expiration date (TOD
*                          format)
SOVSVOLN   DS    F       Binary value indicating # of volumes
SOVSVOLS   DS    CL6     Volume name. This field is repeated
*                          SOVSVOLN times
SOVSCLTN   DS    F       Binary value indicating # of clients
*
*  To work out the address of the start of the following area, add SOLVSVOLN *
*   L'SOVSVOLS + L'SOVSCLTN rounded up to the next fullword to the address of SOVSVOLS.
*
*  The following area occurs SOVSCLTN times unless LOPT has not specified Clients
*  either explicitly or implicitly.
*
SOVSCLNT   DS    OCL120  Client record.
SOVSCDSN   DS    CL44    Client DSN
SOVSJNAM   DS    CL8     Creating Jobname
SOVSSNAM   DS    CL8     Creating Stepname
SOVSPNAM   DS    CL8     Creating Procedure name
SOVSOUID   DS    CL8     Owning Userid
SOVSREFM   DS    CL4     Record format
SOVSBKSZ   DS    F       Binary value representing Block size
SOVSLREL   DS    F       Binary value representing Record len
SOVSUNIQ   DS    2F      Uniquifier value
SOVSPGEN   DS    F       Unsigned binary Plus generation value
SOVSMGEN   DS    F       Unsigned binary Minus Generation
*                          value
SOVSSGEN   DS    F       Unsigned binary SAVEGENS
SOVSEXPD   DS    2F      Expiry Date (TOD Format)
SOVSRES2   DS    F       Reserved for expansion.
```

Figure 43. StreamForClient data format

Since it is possible for StreamForClient data to exceed the maximum blocksize for the record, a binary field incremented by 1 from zero (the SOVSSEQN field), forms part of the record key. Where the record size exceeds the maximum blocksize, ExHPDM increments the SOVSSEQN by 1 and continues the record in the next block. If you specify LOPT(ALLEX Clients), ExHPDM does not write Client–related data specified in the SOVSCLNT data structure.

Example 2

The following example lists all the stream information for the client datasets PROD.*.BACKUP in ClientForStream mode. The data is directed to the CLNTOUT DDname. A description of the resulting data follows.

```
ADMIN DATABASE LIST CLIENT(DSN(PROD.*.BACKUP)) FORMAT(CLIENTFOR
STREAM) OUTDD(CLNTOUT)
```

```
SOVCSRDW    DS    F      Length of data (excluding RDW)
SOVCFSID    DS    CL4    Eyecatcher:  " CFS"
SOVCDSN     DS    CL44   Client DSN
SOVCJNAM    DS    CL8    Jobname
SOVCSNAM    DS    CL8    Stepname
SOVCPNAM    DS    CL8    Procedure name
SOVCOUID    DS    CL8    Owning Userid
SOVCREFM    DS    CL4    Record format
SOVCBKSZ    DS    F      Binary value representing block size
SOVCLREL    DS    F      Binary value representing record length
SOVCUNIQ    DS    2F     Uniquifier value
SOVCPGEN    DS    F      Unsigned binary Plus generation value
SOVCMGEN    DS    F      Unsigned binary Minus Generation
*                          value
SOVCSGEN    DS    F      Unsigned binary SAVEGENS
SOVCEXPD    DS    2F     Expiry date (TOD format)
SOVCVOLN    DS    F      Binary value indicating # of volumes
*
*  The following data is not written if LOPT does not have Streams specified
*  either explicitly or implicity.
*
SOVCNAM     DS    CL8    Stream file
SOVCFNAM    DS    CL44   Stream file name
SOVCDATE    DS    2F     Date stream created (TOD format)
SOVCDEVT    DS    CL4    Device Type
SOVCESOT    DS    CL8    Esoteric name
SOVCCLTN    DS    F      Binary value indicating # of clients
SOVCUBLK    DS    F      Binary value # of blocks used
SOVCTBLK    DS    F      Binary value Total number of blocks
SOVCEXPT    DS    2F     Expiration date (TOD format)
SOVCCNT     DS    F      # of volume for this stream
SOVCVOLS    DS    CL6    Volume name. This field is repeated
*                          SOVCCNT times
```

Figure 44. ClientForStream data format

If you specify LOPT(ALLEX Streams), ExHPDM does not write Stream–related data.

Example 3

The following example list all the ExHPDM streams with a stream name of STREAM1 including all the Client Generations and produces the database report format as ExHPDM V2.0 release:

```
ADMIN DATABASE LIST FORMAT REPORT(V20) STREAM(STREAM1) GEN(ALL)
```

Database BACKUP and RESTORE Commands

BACKUP of the ExHPDM database is necessary not only to ensure that the database can be recovered in case of media or software failure but also to maintain database performance and clean up obsolete records.

Because the ExHPDM database must be shared among multiple systems, updating of the database must be done with the proper interlocking. To be efficient, all transactions on the database need to be performed in small chunks.

ExHPDM deletes a client file by adding a record that indicates that the client file is no longer available. Thus, database size increases at the time the user deletes a client file. Actual deletion of client file records is postponed until the process will not impact other operations.

Because actual deletion of client generations requires extensive change to the database, it can only be performed when the database is temporarily inaccessible to other systems. The only opportunity for this is at backup or restore time, since use of these commands requires the database to be quiesced to all systems other than the one being used to perform the backup or restore operation.

ExHPDM takes advantage of the database backup window to clean up the database. First, ExHPDM performs the backup. Then, if all goes well, it cleans up the client records by either deleting them completely or shrinking them. ExHPDM notes at the end of the backup log any client records compressed in this way. When these processes are completed, ExHPDM brings the database back online ready for normal operation.

Database RESTORE Processing

The ADMIN DataBase RESTORE command restores the database from a previous backup. Before submitting this job, you should prepare the ExHPDM system(s) that are sharing the database as follows:

1. Set up the startup parameters for the ExHPDM system that is to be used for the restore process in the same way that it was set up for backup; that is, the **BACKupDSN** and **BACKupJouRNaL** prefixes specified on the DATABASE keyword should be the same as they were on the last applicable backup.
2. Make sure that the database and journal backup files reside on a DASD that is accessible from the system that performs the restore.
3. Prior to running the restore, delete and re-allocate the current database and journal.The database and journal must be empty at the beginning of the process. Starting any ExHPDM system against the database does, by default, insert formatting records into the empty database. Since this is not appropriate for the restore operation, you must use the following procedures to ensure the database remains empty:
    - If ExHPDM is running and a restore is necessary, issue a SET **DB QUIESCE** command. Refer to “SET Command” on page 200. When the system is

quiesced, submit an IDCAMS delete/redefine for the database (see SAMPLIB member SOVDBDEF).

or

- If ExHPDM is shutdown and a restore is necessary, edit the startup parameter file to add the **QUIESCE** parameter and start ExHPDM. Refer to "DATABASE Keyword" on page 102. This ensures that ExHPDM starts in a quiesced mode so the database is not accessed. The IDCAMS delete/redefine can be performed before or after ExHPDM is started.

or

- If ExHPDM is shutdown and a restore is necessary, start ExHPDM using the **QUIESCE** startup option. Refer to "Chapter 7. Starting the ExHPDM Server" on page 175. This ensures that ExHPDM starts in a quiesced mode so the database is not accessed. The IDCAMS delete/redefine can be performed before or after ExHPDM is started.

4. All ExHPDM systems must have their databases quiesced, including the system that performs the restore. Preferably, the other systems should not even be started. The system that performs the restore, however, must be up and running.

After a restore operation is successfully completed, issue the SET **DB RESTART** command to all applicable ExHPDM systems.

A database can be restored directly from a particular data set backup version (with no journals applied) if the database can be restored by applying journals to an older backup version.

If journals are being applied sequentially, a missing journal backup version is considered to be a null journal. This may or may not be what you intend. You should examine the output from the restore job to ensure all relevant data is applied.

Hint: If possible, you should use the same system to perform the restore that performed the backup. Also, you should consistently either use journaling or not use journaling. Switching between the two can cause missing data.

Keep the names of all files and prefixes unchanged to ensure that ExHPDM does not lose track of old backup data. If you do change the names of files take a backup at your earliest opportunity. If you change the backup prefix, previous backups are hidden from ExHPDM. You must either manually delete them, if they are not required, or rename them to be consistent with the new backup prefix.

Database BACKUP Processing

For each ADMIN DATABASE BACKUP command to be successful, make sure the following conditions are met before submitting the command for backup:

- If you are using journaling, you must quiesce the database on all systems that share the database except the system that performs the backup.

- The system that performs the backup must specify valid backup prefixes in its startup parameter file.
- If you use journaling, all journal files must be readable from the system that performs the backup.

After a you successfully complete a backup operation, issue the SET **DATABASE RESTART** operator command to all applicable ExHPDM systems (except the system that performed the backup, which remains active).

Hint: The same ExHPDM system should always be used to perform backups. This is advisable because the catalog entries for all previous backups need to be accessible from the system performing the backup. If this is not the case, some previous backups may not be in the catalog, which causes ExHPDM to assign an incorrect backup version number.

Should the backup process detect that one or more journals are not closed, the backup cannot proceed until the operator replies to a query. The prompt to the operator states the three alternatives that are available:

1. Cancel the entire backup.
2. Skip the offending journal.
3. Proceed as if the journal was closed.

The recommended procedure is to cancel the entire backup. Review the status of all the ExHPDM systems and correct the ones in error and resubmit the backup.

Skipping the offending journal may be used if the offending journal contains unimportant data (for example, data from a test system).

Proceeding as if the journal was closed may be used when the journal was closed (using SET **DATABASE QUIESCE** operator command) so the backup can proceed. This condition might occur when an ExHPDM system was shut down abnormally (for example, cancelled) and the ExHPDM system was unable to update the journal status record in the database. In this case, the backup can proceed safely using the reply (3) so long as the operator is certain that the journal is indeed closed.

Note: Note that if a journal has no data, it is not involved in the backup process. The backup process can only determine that a journal is empty if it is closed.

The backup process proceeds as follows. If you are not doing journal backups, ExHPDM omits the steps marked with '*****'.

1. Determine the database backup prefix (**BACKupDSN** parameter of the startup parameter file DATABASE keyword). If there is no database backup prefix, the backups cannot be performed.
2. Determine the journal backup prefix (**BACKupJouRNaL** parameter of the startup parameter file DATABASE keyword). If there is none, proceed with the remaining steps, omitting steps marked '*'.

3. Ensure that the database is not quiesced on the system from which you are processing the backup. If it is quiesced on that system, terminate the backup.
4. *Read journal records from the database. These records indicate the DSNs and open or close status of all applicable journals.
5. *For each journal that is not closed, prompt the operator to issue the SET **DATABASE QUIESCE** operator command to close the journal.
6. Issue an IDCAMS LISTCAT LVL on the database backup prefix (* and journal prefix, if applicable). This allows the backup process to assign the next sequential backup version number for the current backup. The version number is constructed as 'V*nnnnnnn*', where *nnnnnnn* is a 7 digit decimal number, starting at 0000001 for the very first backup. The backup process looks for the highest existing version number, then adds 1.
7. *Close this system's journal so it may be copied.
8. *Create a backup journal data set that contains a consolidated copy of all journals determined by steps 4 and 5. The size of this data set is determined by the total size of all journals whose size can be determined, plus a secondary allocation of 1/4 its size in case there are journals whose size could not be determined.
9. *For each journal, open it and copy its contents into the backup journal. When copying is complete, reset the journal by rewriting its first block. If the journal cannot be opened, prompt the operator for an appropriate action:
    - Cancel the backup process
    - Skip the journal
    - Try reopening the journal (after some independent operator action to make the journal accessible).
10. *When all journals are copied, write the header block in the backup journal. This block contains information about all the journals saved in the backup. Because of this seek and overwrite, journal backups must be on a DASD.
11. *Reopen this system's journal.
12. Allocate a database backup file. Allocation parameters are LRECL of 1024 and block size of 6144, with variable blocked RECFM. Primary allocation is sufficient for the contents of the database as determined by subtracting the available free space from the high used RBA. Secondary allocation is 1/4 of this, but should not actually use any of this.
13. The database contents are read sequentially and copied to the backup.

The IDCAMS LISTCAT LVL that was issued in Step 6. is used to create backup version numbers. If you never delete backup versions, there are backup files for versions 1 through n. The next backup version is n+1. The corresponding journal back ups, if used, are also versions 1 to n, with n+1 reflecting the next journal backup. It is possible for holes to exist in either database or journal backups, if the operator or system programmer deleted some or if there was no journal data for some of the backups.

If 0 is less than or equal to version m and version m is less that version n, the state of the database may be restored to version n either by using backup version n with no journals applied, or by starting with version m and applying journals m+1, m+2,... n–1, n in sequence.

There is always an empty backup 'version zero', although no physical file exists for it. This allows version 1 of the database to be obtained either from database backup version 1 directly, or by applying journal backup version 1 to version zero of the database (an empty database).

Syntax

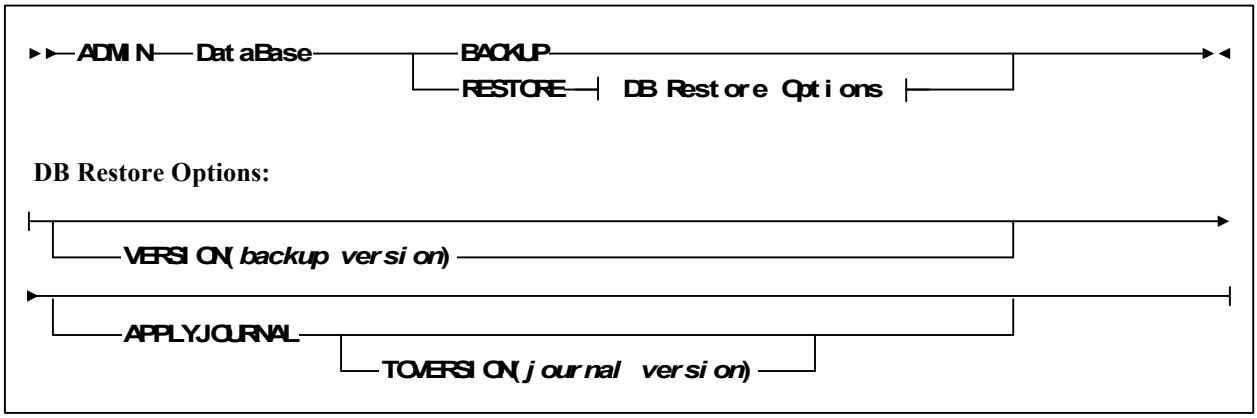


Figure 45. Database Backup Admin Diagram

DataBase BACKUP Parameter

BACKUP
has no parameters.

Note: If ExHPDM is started in disaster recovery mode, this command is not supported.

DataBase RESTORE Parameters

VERSION
backup version specifies the backup version to be used in the restore. If VERSION is not specified, the latest (highest V*nnnnnnn*) cataloged backup version is used.

APPLYJOURNAL
specifies that journals are to be applied during the restore. If APPLYJOURNAL is not specified, then no journals are applied.

TOVERSION
journal version specifies the highest journal version that the restore is to use. Journals are applied from the next highest cataloged version from the specified, or defaulted, backup version up to the highest journal version. If TOVERSION is not specified, the highest journal version is used.

Note: If ExHPDM is started in disaster recovery mode, this command is not supported.

Examples

Example 1

The following example backs up the ExHPDM database. ExHPDM adds the backup version number to the suffix of the back up database and journal names. See “DATABASE Keyword” on page 102 for details about the database back up file names. The backup results in the journal being cleared. No ExHPDM updates to the database are permitted during the backup processing. All database activity waits for the backup to complete. If the database is being shared, all ExHPDM systems other than the one used to perform the back up must have their database in a quiesced state. If not, messages are issued to the operator. The operator has the opportunity to quiesce the other system(s), cancel the operation, or bypass the journal information stored by those systems

```
ADMIN DATABASE BACKUP
```

Example 2

The following example restores the ExHPDM database from the last back up version (that is, the highest version cataloged in the ICF catalog) and applies all journals to the database up to the current backup version number. The database must be quiesced whether you are using the SET **DATABASE QUIESCE** operator command or the ExHPDM **STARTUP** commands. The restore is not allowed until the database is quiesced. The database must be empty; that is, it must be deleted and redefined using IDCAMS. Access to the ExHPDM database is not allowed until a SET **DATABASE RESTART** command is issued.

Note: One of the first steps in the restore process is to make a backup of the current journal(s). This backup is identical to the normal backup process except that the database itself is not backed up. Thus a new generation of journal backup is created without a corresponding database backup generation.

```
ADMIN DATABASE RESTORE APPLYJOURNAL
```

Example 3

The following example restores the ExHPDM database from backup version 5 and applies journals up to version 7. Any journals from version 7 and higher are ignored. The database must be quiesced, either using the SET **DATABASE QUIESCE** operator command or the ExHPDM startup parameter file keywords.

```
ADMIN DATABASE RESTORE VERSION(5) APPLYJOURNAL
TOVERSION(7)
```

Other Administration Commands

Commands described in this section are:

- SCANSTReamfile and Disaster Recovery Mode

- SCANEXPIRE.

SCANSTReamfile and Disaster Recovery Mode

The SCANSTReamfile command provides a method of accessing and restoring client data sets from a known stream file when the ExHPDM database is inaccessible or not available. The ExHPDM server runs in a restricted manner that only reads stream files using existing client JCL to restore the data sets. No changes are needed to the client JCL.

Use the SCANSTReamfile command only in a disaster recovery situation; it should be used as a last resort to recover client data sets from a known stream file when the ExHPDM database cannot be reinstated.

Note: Any client files restored with the SCANSTReamfile command are not saved in any external database. That is, when the ExHPDM database is reinstated, it does not necessarily contain information about the data sets restored by the SCANSTReamfile command.

In disaster recovery mode there is no external database in which to store client file information. Instead, the SCANSTReamfile command stores the client database entries into a database in memory called the volatile database.

Should the ExHPDM server be terminated at any time while operating in disaster recovery mode, all information in the volatile database is lost.This means that the corresponding stream files of any incomplete client restore jobs or connections must undergo SCANSTReamfile command processing again to reestablish their entries in the volatile database.

The optimum use of the SCANSTReamfile command is to recover either a copy of the ExHPDM database itself or a copy of a backed up ExHPDM database.

Using SCANSTReamfile

ExHPDM supports the SCANSTReamfile command is only when an ExHPDM server is started with the **DR** parameter.

When you start ExHPDM with the **DR** parameter specified, ExHPDM builds a dummy parameter list, internally, to simulate the environment for internal processing. ExHPDM does **not** open the STKPARMS ddname to read its startup parameter file.

Note: When you issue the SCANSTReamfile command, you must know the name of the volume or volumes that contain the data set to be restored and the tape devices on which to mount the volumes. You must get this information from an ADMIN DATABASE LIST command issued **before** the need for a disaster recovery operation, when the ExHPDM database was available.

ExHPDM starts stream task each time you issue a SCANSTReamfile command. The stream task scans the stream file, validates the volume serial number sequence and collects information about each client data set on the stream file. When the process reaches the EOF for each stream file in the set, ExHPDM closes the stream file and dismounts the tape volume(s).

ExHPDM uses the information it collects about each client data set from each stream file set to build a temporary volatile database image in memory. This volatile database is available only as long as the ExHPDM server is in disaster recovery mode. When ExHPDM is shutdown, the volatile database is no longer available.

The information ExHPDM collects about the stream file contents includes:

- stream file name
- stream file volume serial numbers
- client data set names and uniquifiers
- client data set record count, logical record length, record format, and block size
- client data set 256K Byte block count
- start and end volume sequences and block identification
- date and time when each client data set was written.

You access the volatile database using the same procedures as for normal mode:

- Use the ADMIN DATABASE LIST command to access the information in the volatile database.
- Use client restore jobs to access the client data sets for which the SCANSTReamfile command has built the corresponding volatile database records.

ExHPDM in Disaster Recovery Mode

ExHPDM operates in disaster recovery mode to perform SCANSTReamfile command processing. Use the MVS START operator command **DR** parameter to specify disaster recovery mode in the ExHPDM server when it is started. For example:

```
S ExHPDM,PARM='SSNAME(SOVW) DR'
```

In disaster recovery mode, you have no access to any external VSAM database. Also, the PRM or SOVPRM parameters of the MVS START operator command do not specify a parmlib member, since the process uses default internal parameters.

The following example shows the messages that appear when ExHPDM is started in disaster recovery mode:

```
22.55.59 STC00705 $HASP373 SOVWT STARTED
22.55.59 STC00705 IEF403I SOVWT - STARTED - TIME=22.55.59
22.56.01 STC00705 SOV03025I SSI startup parameters parsed successfully.
22.56.01 STC00705 SOV03094W ExHPDM subsystem SOVW is operating in disaster recovery
mode.
22.56.03 STC00705 SOV05007I Log Messages are being directed to the MVS console with routing
code 11.
22.56.03 STC00705 SOV03001I ExHPDM 2.0.0 is starting.
22.56.03 STC00705 SOV09106I Database manager started in quiesced mode.
22.56.03 STC00705 SOV06034I Stream SCANSF will use the internal default device definition
DEFAULT.
22.56.03 STC00705 SOV03069I Operator interface active. ExHPDM command recognition string is
SOVW.
22.56.04 STC00705 SOV03053I ExHPDM is waiting for work.
```

Unsupported Commands in Disaster Recovery Mode

When you start the ExHPDM server in diaster recovery mode, ExHPDM does not support some parameters from the Administration Utility that operate on the real database and certain operator commands. For example, in disaster recovery mode ExHPDM ignores the STKPARMS ddname.

The following ADMIN DATABASE commands are not supported when the ExHPDM server is started in disaster recovery mode:

- **RESTORE**
- **BACKUP**.

If you specify these parameters in disaster recovery mode, they are flagged as errors.

The following operator commands are not supported when the ExHPDM server is started in disaster recovery mode:

- **SET PRM**
- **SET DATABASE.**

If you specify these parameters in diaster recovery mode, they are flagged as errors.

The following client DD SUBSYS parameters are not supported when the ExHPDM server is started in disaster recovery mode:

- **WAIT(YES)**
- **CLASS.**

If you specify these parameters in disaster recovery mode, they are ignored.

When the ExHPDM server is started in disaster recovery mode, ExHPDM will not write or dump output to stream files. No new stream files can be created in disaster recovery mode. If you request these operations, they are flagged as errors.

When the ExHPDM server is started in disaster recovery mode, it logs an informational alert message to the JESMSG data set and the ExHPDM server log for all client connections indicating the mode. The ExHPDM DISPLAY command also indicates that displays are in disaster recovery mode.

Command Considerations in Disaster Recovery Mode

Be aware of the following considerations when you use the SCANSTReamfile command.

Bypass Label Processing

The SCANSTReamfile command processes the selected volumes in Bypass Label Processing (BLP) mode. In many installations Bypass Label Processing mode is a protected resource because such processing bypasses conventional tape label processing and opens up the possibility of a security breach.

Bypass Label Processing is usually protected by the installation's MVS security system.

The ExHPDM server needs access to the Bypass Label Processing resource so it can process tape volumes for the SCANSTReamfile command. Bypass Label Processing permits the SCANSTReamfile command tape I/O processes to examine all data blocks and tape marks on a tape volume. In addition, it provides a means for the SCANSTReamfile command to successfully process a tape volume without knowing its data set name.

An installation with a tape management system installed may experience additional prompting for a volume serial number when the SCANSTReamfile command processes each tape volume. The following is an example of a CA–1 tape management system prompting for a volume serial number for Bypass Label Processing:

```
22.56.19 STC00705 SOV06211I SCANSF SCANSTReamfile stream task (ID 6) started.

22.56.24 STC00705 *IEC501A M 0B61,105706,BLP,COMP,
SOVWT,SOVWT,SYS98125.T225619.RA000.SOVWT.R0100236

22.56.24 STC00705 *TMS008 IEC501A M 0B61,105706,BLP,COMP,
SOVWT,SOVWT,SYS98125.T225619.RA000.SOVWT.R0100236

22.56.38 STC00705 *0060 IECTMS1 0B61,     ,ENTER VSN
```

Effective Stream File Name

The SCANSTReamfile command constructs an internal name for the stream file that consists of the string SOV and the name found in the tape volume header 1 HDR1 label. SOV is added as a prefix to the 17 characters of the stream file name that is stored in the HDR1 label to form the temporary stream file name in the volatile database.

Volume Processing Sequence

The SCANSTReamfile command processes the selected volumes in the order specified by its **VOLUMES** parameter. You must take care to ensure the correct order of volumes for the stream file.

The SCANSTReamfile command verifies the sequence of ExHPDM data blocks on the tape volumes and rejects the SCANSTReamfile command process if the data blocks are out of order. If ExHPDM rejects the SCANSTReamfile command process, it is necessary to run a SOVADMN ADMIN DATABASE DELETE STREAM(* VOL(volser...)) to remove the partial stream file from the in-memory volatile database.

The SCANSTReamfile command completes when it detects an SL EOF1 block.

Processing Incomplete Stream File Volumes

The SCANSTReamfile command attempts to process the contents of each tape volume by examining data for standard label information, such as HDR1, HDR2, EOF1, and EOV data blocks. It also examines blocks of data that are not in ExHPDM format.

The SCANSTReamfile command terminates processing when it cannot determine the data structure of the volume or when it determines that a tape volume does not belong to the data set found in the first tape volume's HDR1 label.

If SCANSTReamfile processing processes and rejects an erroneous non-stream file volume, that volume cannot be deleted through the SOVADMN DATABASE DELETE command because it has no stream or client file records. If the volume serial number presents a problem, the volume can be removed only by restarting the server address space. This requires that you redo all previous SCANSTReamfile processing.

Stream File Volumes Already Scanned

If the SCANSTReamfile command scans a volume it has already scanned, it issues a message that is has already scanned or processed the volume. It terminates the client and issues the following message:

SOV06235W Volser '105706' has already been scanned - it belongs to stream file 'SOV.L.STRM01.AOBHLAP1'.

SOV06238E One or more volsers selected for this SCANSTReamfile operation have already been scanned. Operation terminated.

Processing Non–Stream File Volumes

The SCANSTReamfile command attempts to diagnose whether or not a foreign tape volume contains a valid stream file. If no valid stream file is found, SCANSTReamfile processing terminates.

However, in some cases where EOT is encountered, as in the case of an uninitialized tape, and no data is present, the SCANSTReamfile command may encounter a physical I/O error. This I/O error is processed by MVS IOS tape error recovery to yield an IOS000I

message and possibly to request a swap operation to another drive. In this case, the stream task must be cancelled with either the CANCEL STREAM(stream name) command or the CANCEL STREAM(stream id) command. This terminates the stream task and exits MVS swap processing. It also dismounts the offending tape volume.

The following is an example of cancelling a stream task named SCANSF when the swap request is rejected:

```
SOV06241I SCANSTReamfile: scanning streamfile SOV.#00000000000000000.
SOV06242I SCANSTReamfile: processing volume 400023 (1 of 10).
   - Please wait...
IOS000I 0B60,9D,EOD,02,0E40,,**,400023,SOVSH
   08402036400003200B02(371D35F0000001FF)FFF0(01400003)CE081B1003810000
IOS000I 0B60,9C,VOI,02,0E40,,**,400023,SOVSH
      08402031400003200600(3701492535420000)0000(00000000)CE081B1003810000
=SLS0310I Swap will be automated
*IGF500I  SWAP 0B60 TO 0B61 - I/O ERROR
*0062 IGF500D  REPLY 'YES', DEVICE, OR 'NO'
r 62,no
IEE600I REPLY TO 0062 IS;NO

sovs cancel stream scansf

SOV02103I ExHPDM command issued.
SOV06401I All tasks for stream SCANSF cancelled.
SOV06208E Client STC:SOVSH.SOVSH.ExHPDM, DSN DISASTER.RECOVERY.MODE:
   -  terminated due to CANCEL STREAM command.
TMS014  IEC502E K 0B60,400023,,SOVSH,SOVSH,
      SYS98128.T011412.RA000.SOVSH.R0100167
SOV062141 SCANSF stream task (ID 41) ended.
```

Hint: When more than one stream file is to be read or recovered using the SCANSTReamfile command, it is best to perform the SCANSTReamfile followed immediately by the client restore jobs for that stream file. This minimizes the possibility that a SCANSTReamfile will need to be redone in the case the volatile database becomes corrupted because of incorrect volume serial numbers being provided.

After the operation of each SCANSTReamfile command, you should run an ADMIN DATABASE LIST command of the volatile database to provide feedback about all the valid client data sets that were retrieved from the stream volumes.

Note: The SCANSTREAMFILE can take a considerable amount of time to process, particularly when large capacity tape cartridges are being scanned. Therefore, it is recommended that you submit the SCANSTREAMFILE with a TIME=parameter of 1440 to avoid any 522 abends.

Syntax

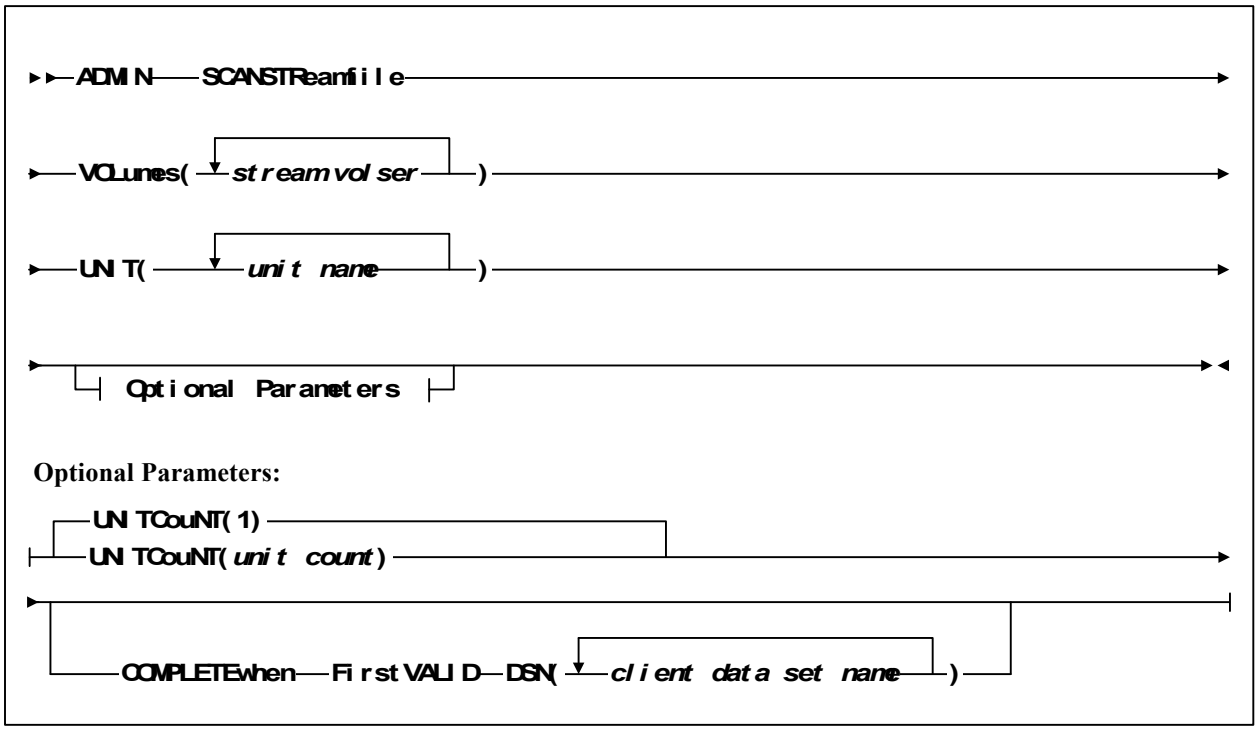


Figure 46. ADMIN SCANSTReamfile Command

Command Name

The ADMIN SCANSTReamfile command starts a data set disaster recovery command.

Parameters

VOLumes
specifies one or more tape volume serial numbers that comprise a known stream file. The volume serial numbers must be in the correct sequence.

UNIT
specifies a list of one or more tape device specifications as generic or esoteric name.

UNITCouNT
specifies the number of devices to be used to satisfy the device list request specified in the **UNIT** parameter.

UNITCouNT(1) is the default.

COMPLETEwhen
specifies that the stream file scanning is to stop when a requested event has completed.

FirstVALID

specifies that when the specified condition is the first valid, stream file scanning is to cease.

DSN

specifies a list of one or more fully qualified client data set names. No masking is permitted. The SCANSTReamfile command completes when at least one valid occurrence of the listed client data set names is located.

Note: The ADMIN DATABASE LIST command may be used to inspect the contents of the volatile database at any time. After the operation of each SCANSTReamfile command, you should run an ADMIN DATABASE LIST command of the volatile database to provide feedback about all the valid client data sets that were retrieved from the stream volumes.

Examples

The following are examples of the SCANSTReamfile command, including an example of the ADMIN DATABASE LIST command and an example of a SOVDSSU RESTORE job.

Example 1

In the following example, all the client data sets on a stream file (whose volume serial numbers are BACK01, BACK02, and BACK09) are scanned in data recovery mode so client restore jobs can be run against them. The SOVADMN jobstep is to connect to a server's subsystem named SOVW. When this job completes, standard client batch jobs can be run against the ExHPDM server to retrieve client files. See Example 4 for an example of a standard job to restore files while in data recovery mode.

```
//ADMINJOB  JOB jobcard info
//*
//*  SCANSTReamfile - build volatile database for
//*                   for these stream files.
//*
//SCANSTR  EXEC  PGM=SOVADMN
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD SYSOUT=*
//SNAMSOVW DD DUMMY         /*use SOVW ExHPDM server*/
//SYSIN    DD *
ADMIN SCANSTReamfile VOL(BACK01 BACK02 BACK09)
                   UNIT(CART)
ADMIN DATABASE LIST
/*
```

Example 2

In the following job step example, a database listing is obtained while the ExHPDM server is in disaster recovery mode. Note that there is no change to the JCL used when the ExHPDM server is not in disaster recovery mode.

```
//ADMINJOB  JOB jobcard info
//*
//*  SOVADMN- DATABASE LIST Utility
//*
//LISTDB  EXEC PGM=SOVADMN
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD  SYSOUT=*
//SNAMSOVW DD  DUMMY         /*use SOVW ExHPDM server */
//SYSIN    DD *
ADMIN DATABASE LIST CLIENT (DSN(**) GEN(ALL))
/*
```

Example 3

In the following example, all the client data sets on a stream file (whose volume serial numbers are BACK01, BACK02, BACK09, TIM023, and EVT991) are scanned with the ExHPDM server in disaster recovery mode so client restore jobs can be run against them.

Two tape devices are allocated to lessen the impact of mount times.

Because only certain client data sets are required, scanning is terminated when the first intact/complete copies of all listed client data set names are found.

```
//ADMINJOB  JOB jobcard info
//*
//*  SCANSTReamfile - build volatile database for
//*                   for these stream files.
//*
//SCANSTR  EXEC  PGM=SOVADMN
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT DD SYSOUT=*
//SNAMSOVW DD DUMMY         /*use SOVW ExHPDM server*/
//SYSIN    DD *
ADMIN SCANSTReamfile
VOL(BACK01 BACK02 BACK09 TIM023 EVT991)
UNIT(3490)
UNITCNT(2)
COMPLETEwhen
FVALID
DSN(
IMSPROD.DATABASE203.BACKUP
STKECC.CATMS1.BACKUP
LAST.DITCH.TSO.PROCLIB
)
ADMIN DATABASE LIST CLIENT(DSN(**) GEN(ALL))
/*
```

Example 4

The following is an example of a standard DFSMSdss SOVDSSU (ADRDSSU) job step to restore two files while the ExHPDM server is in disaster recovery mode. The ExHPDM server is started in disaster recovery mode and the previous jobs are run successfully.

Note that the job has no special parameters for disaster recovery mode processing. The connections in the client process use the information in the volatile database in the main storage of the ExHPDM server that was built by previous runs of the ADMIN SCANSTReamfile command.

The ExHPDM server allocates tape devices using the device name used in the ADMIN SCANSTReamfile **UNIT** command for this stream file.

Additionally, the data set PRODCICS.MAININDX is to be restored from a full volume backup dump of a GDG data set named PRODCICS.FULLBKUP.G0123V00. The explicit GDG data set name was found as the result of a ADMIN DATABASE **LIST CLIENT(DSN(**))** command (see Example 2) that was run against the ExHPDM server while it was in disaster recovery mode after the SCANSTReamfile command was run.

```
//RESTJOB  JOB jobcard info
//*
//*  Normal DF/DSS restore data sets job.
//*
//RESTORE    EXEC PGM=SOVDSSU
//STEPLIB  DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT   DD SYSOUT=*
//IMS001       DD DSN=PRODBKUP.IMS001,DISP=OLD,SUBSYS=SOVW
//TSO998   DD DSN=PRODBKUP.TSO998,DISP=OLD,SUBSYS=SOVW
//CICSPROD DD DSN=PRODCICS.FULLBKUP.G0123V00,DISP=OLD,
//                 SUBSYS=SOVW
//SYSIN        DD *
RESTORE DATASET(INCLUDE(IMSPROD.DATABASE203.BACKUP)) -
                OUTDYN(IMS001) INDD(IMSFILE)    -
                TOLERATE(ENQFAILURE)
RESTORE DATASET(INCLUDE(LAST.DITCH.TSO.PROCLIB)) -
                OUTDYN(TSOSMS) INDD(TSO998)     -
                TOLERATE(ENQFAILURE)
RESTORE DATASET(INCLUDE(PRODCICS.MAININDX))     -
                OUTDYN(CICS43) INDD(CICSPROD)   -
                TOLERATE(ENQFAILURE)
/*
```

Example 5

The following is an example of performing several SCANSTReamfile commands, one of which specifies the **COMPLETEwhen** parameter. The scan is to complete when it finds the first complete (valid) client file named OWWT.BACKUP.DELETE.OWWTCP. It then scans all client files from the volume 105706. This example also performs a database list to examine what is in the volatile database.

```
//ADMINJOB  JOB jobcard info
//* SCANSTReamfile
//SCANSF          EXEC PGM=SOVADMN
//STEPLIB   DD  DISP=SHR,DSN=EXHPDM.LOAD.LIBRARY
//SYSPRINT        DD SYSOUT=*
//SYSTERM         DD SYSOUT=*
//SNAMSOVS        DD DUMMY
//SYSIN           DD *
ADMIN SCANSTReamfile
VOL(542571,542573,542627,542629,542631,542632)
  UNIT(CART) UNITCOUNT(2)
COMPLETEWHEN(FVALID(DSN(QWWT.BACKUP.DELETE.OWWTCPT)))
ADMIN SCANSTReamfile VOL(105706) UNIT(CART)
ADMIN DATABASE LIST
/*
```

SCANEXPIRE

The ADMIN SCANEXPIRE command is used to process the ExHPDM database to delete client files which have expired. If you are using client file expiration, whether from ManaGeMenT parameters, by specifying RETPD or EXPDT on the client JCL DD, or by specifying RETainPerioD on the STREAM definition, you must run a SCANEXPIRE job at regular intervals.

If you never run SCANEXPIRE you will not realize the administrative advantages of using client file expiration.

SCANEXPIRE is a convenient abbreviation for the equivalent DataBase DELETE command. The effect of running SCANEXPIRE is to internally generate the following command:

ADMIN DataBase DELETE CLIENT(DSN(**) GEN(ALL) EXPIRED)

The effect of this command is to select all client files (and all generations of those client files) which have expired as of the current date. These client files are deleted from the database. If any stream files are made empty by this process, those stream files are deleted, and any volumes belonging to those stream files are made available for reuse. The TMS is notified via the TMS SCRatchVOLumeRouTine that the volumes are now scratch.

The output from running SCANEXPIRE is exactly the same as the output from running the equivalent DataBase DELETE command.

Syntax

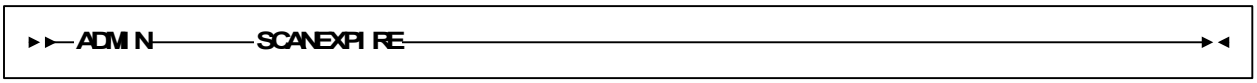


Figure 47. ScanexpireAdmin Diagram

Command Name

The ADMIN SCANEXPIRE command initiates the deletion of expired client data sets.

Parameters

SCANEXPIRE has no parameters. A SCANEXPIRE command is internally converted into a DataBase DELETE command; thus the restrictions which apply to DB DELETE commands also apply to SCANEXPIRE.

Overview of ExHPDM Database Format and Operations

An understanding of how ExHPDM organizes its knowledge base helps to better comprehend the output of the LIST and DELETE commands. This section describes in summary how the database is constructed.

ExHPDM knows about the following types of objects:

- Stream files
- Client files
- Volume serials.

Stream files are the data sets which are physically created whenever one or more client files are written to the ExHPDM subsystem. A stream file is comprised of one or more volume serials. Each stream file contains one or more client files.

ExHPDM manages a set of volumes identified by their volume serial numbers. Each volume serial is represented by a record which contains the volume serial number as a key, with the stream file name as data. This allows ExHPDM to determine the unique stream file that a particular volume is a part of.

Another set of records, given a stream file name, allows ExHPDM to determine the list of volume serials which comprise that stream file.

ExHPDM also manages a set of client files identified by their data set names. As with the volume serial and stream file data records described above, ExHPDM keeps two sets of records, one for managing client-to-stream mapping, and one for stream-to-client mapping.

An additional complexity of managing client file records is that several “generations” of the same client data set name may simultaneously exist in the database.

Normally a user is concerned only with the most recent generation. However, it is possible to access previous generations of a data set by using DD SUBSYS parameters in the read job.

Because more than one generation of a client data set name can exist at the same time, each client file is combines two things: the client data set name and a so called 'uniquifier'. The uniquifier is a unique number derived from the system clock.

Writing Client Files to the Database

The following sequence of database operations occurs when ExHPDM writes a client file:

1. If this client requires that a new stream file be created, that is, if it cannot join an existing write stream, ExHPDM creates the new stream file using the following steps:
    - It allocates a new stream file (with a uniquely generated name). The operating system returns information on the volume serial(s) used. For each volume serial, ExHPDM creates a database entry that refers back to the stream file name.
    - It creates a stream file record that lists all volume serials.
    - It creates a stream file record, initially empty, to list each client file that is added to this stream.
2. When the client file starts being written, ExHPDM makes an entry to the stream file record indicating this client file.

3. When the process of writing the client file successfully completes, ExHPDM adds a new generation of this client to the client DSN record. This entry points back to the stream file that contains this client file. If, on the other hand, the client write process did not complete successfully, ExHPDM does not add anything to the client DSN record. Instead, it amends the stream file record entry (initialized in (Step 2)) to indicate that the client did not complete successfully.

   Note: ExHPDM creates a client generation only for client files which complete successfully, with no errors. If a client file is, for example, cancelled during its write, it is not counted as one of the valid 'generations' of this client DSN.

Reading Client Files from the Database

The sequence of database operations for reading client files from the database is:

1. ExHPDM accesses the client DSN record for the requested generation (usually the latest). This points to the appropriate stream file.
2. Next, ExHPDM accesses the stream file record that contains the list of volume serial numbers.
3. Finally, ExHPDM mounts the appropriate volume serial and starts reading the client file.

Listing Objects in the Database

Because the set of volume serials is relatively small and manageable, and because all stream and client files must reside on a limited and consistent set of these volume serials, ExHPDM generally considers the set of volume serials that it knows about as the ultimate source of information.The database listing process consists of the following steps:

1. ExHPDM obtains the complete list of volume serials under its control.
2. From these, it obtains the set of stream files. Most of these stream files may be discarded, since particular user parameters cause a subset of stream files to be selected.
3. From the stream files it has selected, ExHPDM obtains the client files contained in them. Again, most of the stream and client files may be discarded because of subsetting.
4. An exception to the above processing occurs when particular client DSNs are explicitly selected; that is, when no wildcarding is used. In that case, the above process is extended as follows. For each fully qualified client DSN, ExHPDM obtains the client DSN record directly (without reference to volume serials or stream files). This record contains all extant generations of the client DSN, which allows ExHPDM to construct an additional output listing.

Exceptional Conditions in the Database

Several circumstances beyond ExHPDM's control can interfere with the above procedures. These circumstances all involve the creation of a mismatch between the information that ExHPDM maintains in its database and the 'real world.'

The most common cause of problems is the abuse of 'external data managers' or 'tape management systems'. This includes inadvertent scratching of volumes that are under ExHPDM control.

If a volume under ExHPDM control is scratched without ExHPDM's knowledge, the following situation can arise:

1. The user starts a new client write job.
2. ExHPDM allocates a new stream file. The operating system mounts a scratch volume and presents this volume serial number to ExHPDM.
3. ExHPDM examines its database and discovers that the volume already exists and is part of a previously written stream file.
4. ExHPDM rejects this volume, requesting that another volume be mounted. This is indicated by the following message:

   SOV06213W VOLSER *volume_serial_number* **already used by** *stream_file_name***. Mount another.**

   Even though the rejected volume was a scratch volume, as far as ExHPDM is aware, the volume was part of a previously written stream file.

This situation results in two problems. First, the volume cannot be used by ExHPDM. Second, and more seriously, ExHPDM thinks there is a valid stream file when in fact the stream file may not be accessible because the user scratched one of its volumes.

Another cause of database problems occurs because of the use of BACKUP and RESTORE. RESTORE changes the database to a state that was valid in the past. However, the real world moves on; streams are deleted and created, volumes allocated and scratched, and so on.

When these problems occur, the user must take manual corrective action, judiciously using DELETE commands to remove offending or inconsistent records.

Considerations for Allocating the ExHPDM Database

The Access Method Services (IDCAMS) DEFINE CLUSTER parameters which are optimum for the ExHPDM database critically depend on two factors:

- Whether cross-system sharing is to be used—SHR(4,4) versus SHR(1,3).
- The expected general pattern of use of ExHPDM.

ExHPDM does not normally generate much update I/O to its database, so performance is not usually of much concern. However, some batch-like operations, such as ADMIN BACKUP and ADMIN RESTORE, require extensive update activity. To minimize possible impact to the availability of ExHPDM, you should optimize the database performance as much as is practicable.

If you are not required to share the database across systems it is strongly recommended that you use SHR(1,3). The performance of the database, especially on write or update, is degraded if you specify SHR(4,4), because VSAM is unable to make much use of

buffering, and all writes and all data component reads must access the I/O device rather than buffers in memory. If the database is to be shared, however, you must specify SHR(4,4).

Some IDCAMS options, such as IMBED, are of limited use in a highly cached or virtual DASD environment such as those available on RAIDs. The original intention of IMBED was to reduce seek and/or rotational latency times on a physical disk device. These characteristics of legacy DASD are not applicable to modern I/O subsystems. Indeed, the specification of IMBED in such an environment not only wastes space, but also reduces performance.

Other IDCAMS options relate to VSAM software requirements and therefore are specified independently of the physical environment. Determining the optimum values, for example, CI sizes and freespace, depends, to a certain extent, on how you plan to use ExHPDM. The following IDCAMS DEFINE command is provided as a template for the discussion which follows.

```
DEF CLUSTER -
 (NAME(EXHPDM.DBASE) -
  VOLUMES(AUS002) -
    SHR(4,4)) -
DATA -
 (NAME(EXHPDM.DBASE.DATA) -
 KEYS (63 0) -
 RECORDSIZE(256 1024) -
 FREESPACE(0 80) -
 CYL(80 5) -
 CISZ(7168) -
 BUFFERSPACE(24576)) -
INDEX -
 (NAME(EXHPDM.DBASE.INDEX)
 CISZ(2048))
```

SHR

This should be set to either SHR(1,3) for non-shared ExHPDM database, or SHR(4,4) for cross-system sharing. If sharing is not required, it is strongly recommended that you specify SHR(1,3) , which will allow better performance and space utilization.

KEYS and RECORDSIZE

You must specify KEYS(63 0).ExHPDM will not start with any other value. Moreover, you must set the upper bound of record size to 1024. The average size depends on various factors; however it is not an important parameter, so 256 is a reasonable value.

FREESPACE

This parameter is most critical when you are using SHR(4,4) because it determines how much data can be added to the database before a reorganization becomes

necessary. If you are using SHR(1,3) this parameter is not so critical, as CA splits are permitted, which allows the database to increase in size as required.

It may not be possible to determine the optimum figure when you first install ExHPDM. In this case, approximately 80 percent freespace with an initial allocation of 10 cylinders is a reasonable starting point.

After you have used ExHPDM for some time, you will be able to predict with greater certainty the amount of activity that may be sustained before a reorganization is necessary.

Once you nominate an overall percentage of freespace, you should follow these recommendations for allocating that freespace between CIs and CAs:

- If SHR(1,3) (i.e., nonshared), then divide the freespace so the majority is for CIs. For example, if 80 percent freespace is required, specify

  FREESPACE(75 5).

- If SHR(4,4) is used, then all the freespace should be allocated to the CA and none to the CI, for example,

  FREESPACE(0 80).

In any case, you should not specify more than 90 percent freespace, because of the limited precision with which VSAM initially loads the data set to achieve the requested freespace, given the large record sizes used by ExHPDM.

CYL

If you can choose possible space allocation units (records, tracks or cylinders), it is recommended that you choose cylinders. Allocating the primary (and secondary) space in cylinders creates the largest possible CAs (since each CA is one cylinder).

For SHR(4,4), the secondary space should be between 5 and 10 percent of the primary allocation. This is recommended because the total space requirement may be slightly underestimated during the initial load. This causes one additional extent to be required during the load (for example, at RESTORE time). During subsequent normal operation, however, no additional extents are allocated because of the prohibition on CA splits.

For SHR(1,3), there are no particular recommendations for relative allocation of primary and secondary.

CISZ

A CI size should be specified for both the data and index components of the cluster. The recommended sizes should be used unless your experience dictates the use of some other size. In particular, the index CI size (2048) should not be reduced if SHR(4,4) is being used. This prevents the 'database' (in reality the index component) from appearing to fill prematurely. For example, if the index CI size is reduced (or defaulted) to 1024, the database may appear to fill up before it reaches 50 percent utilization as reported by the DB MONITOR keyword. See “MONITOR Keyword” on page 124 for details on monitoring the database space utilization.

BUFFERSPACE

The default BUFFERSPACE value chosen by IDCAMS is normally satisfactory for the type of processing performed by ExHPDM. However, a slight improvement may be obtained by specifying the given value (24K) which is made up of two data CIs plus 5 index CIs of the recommended size. This allows more of the index set to be buffered in storage, for improved random access. There is little benefit in increasing BUFFERSPACE beyond the given value unless an extremely large (that is, multi-volume) database is in use.

Reorganizing the ExHPDM Database

Eventually the database fills and is unable to expand to accommodate insertions, either because it has run out of extents or space on a volume, or because SHR(4,4) is in effect and a CA split has become necessary. When this happens, you must reorganize the database to make additional free space available.

VSAM administrators may be familiar with the common technique of using the IDCAMS REPRO facility to copy all records out to a 'flat file' then using REPRO to reload the data into a newly reallocated data set. This technique is NOT recommended for ExHPDM. Use ExHPDM BACKUP and RESTORE commands in place of IDCAMS REPRO.

When a reorganization is necessary, the database should be temporarily restarted if it is automatically quiesced. The database must be active to run a BACKUP job. Any client jobs should be placed on hold for the duration of the reorganization process.

Note: Any client job which was abnormally terminated (because the database became inaccessible) must be completely rerun as soon as the database is reorganized. Naturally, it is better to use the MONITOR startup parameter option so that you can anticipate the necessity for reorganization and schedule it for a convenient time, rather than waiting for an emergency caused by running out of space. See “MONITOR Keyword” on page 124 for details on monitoring the database space utilization.

The following steps should be applied to perform the reorganization:

1. Ensure that the database is active to one (and only one) ExHPDM server.
2. Submit an ADMIN BACKUP job as outlined in “Database BACKUP and RESTORE Commands” on page 248.
3. On successful completion of the BACKUP, issue the SET DB QUIESCE operator command as described by the “SET Command” on page 200. This is necessary for the following RESTORE.
4. Delete and redefine the database, with an appropriate (larger) size.
5. Submit an ADMIN RESTORE job as outlined in “Database BACKUP and RESTORE Commands” on page 248.
6. On successful completion of the RESTORE, issue the SET DB RESTART operator command as described by the “SET Command” on page 200

Both BACKUP and RESTORE steps may require some time to complete. A 300-cylinder database may take between one and 20 minutes to restore, depending on system speed and

other workloads. Even starting ExHPDM with a new (empty) database may take a few minutes, since the database needs to be formatted before use.

Chapter 10. DD SUBSYS JCL Parameters

Entering DD SUBSYS JCL

The DD SUBSYS JCL parameters indicates to ExHPDM that a particular client data set are to form a connection to the ExHPDM server. This might be required on both input (read) and output (write) connections.

Options can be specified on the DD SUBSYS JCL parameters to allow for specific requirements of a job. These options override similar parameters that are normally defined in the ExHPDM parameter file. However, where the parameters are explicitly specified in then ExHPDM parameter file, then these override the DD SUBSYS parameter defaults.

See the chapter “About This Book” for the syntax conventions used in this chapter. Note that parenthesis are optional in all ExHPDM DD SUBSYS parameters; however, they are shown in the syntax diagrams and examples in the interest of precision.

An example of coding ExHPDM DD SUBSYS parameters with parenthesis, and the equivalent command coded without parenthesis, and a more readable format, is shown as follows:

```
DD SUBSYS=(SOV,'CATCHUP(WITHIN(BLOCKS(200)))')
DD SUBSYS=(SOV,'CATCHUP WITHIN BLOCKS 200')
```

Parenthesis may sometimes be required to ensure that a command is not ambiguous. An ambiguous command may result if a keyword value happened to be the same as a keyword. For example, if a CLASS name happened to be called CATCHUP, then the following command would require parenthesis:

```
DD SUBSYS=(SOV,'CLASS CATCHUP')
```

In this example, parenthesis are required because CATCHUP is a keyword. The minimum parenthesis required is:

```
DD SUBSYS=(SOV,'CLASS(CATCHUP)')
```

The format of the DD SUBSYS parameters is shown in Figure 48.

Syntax

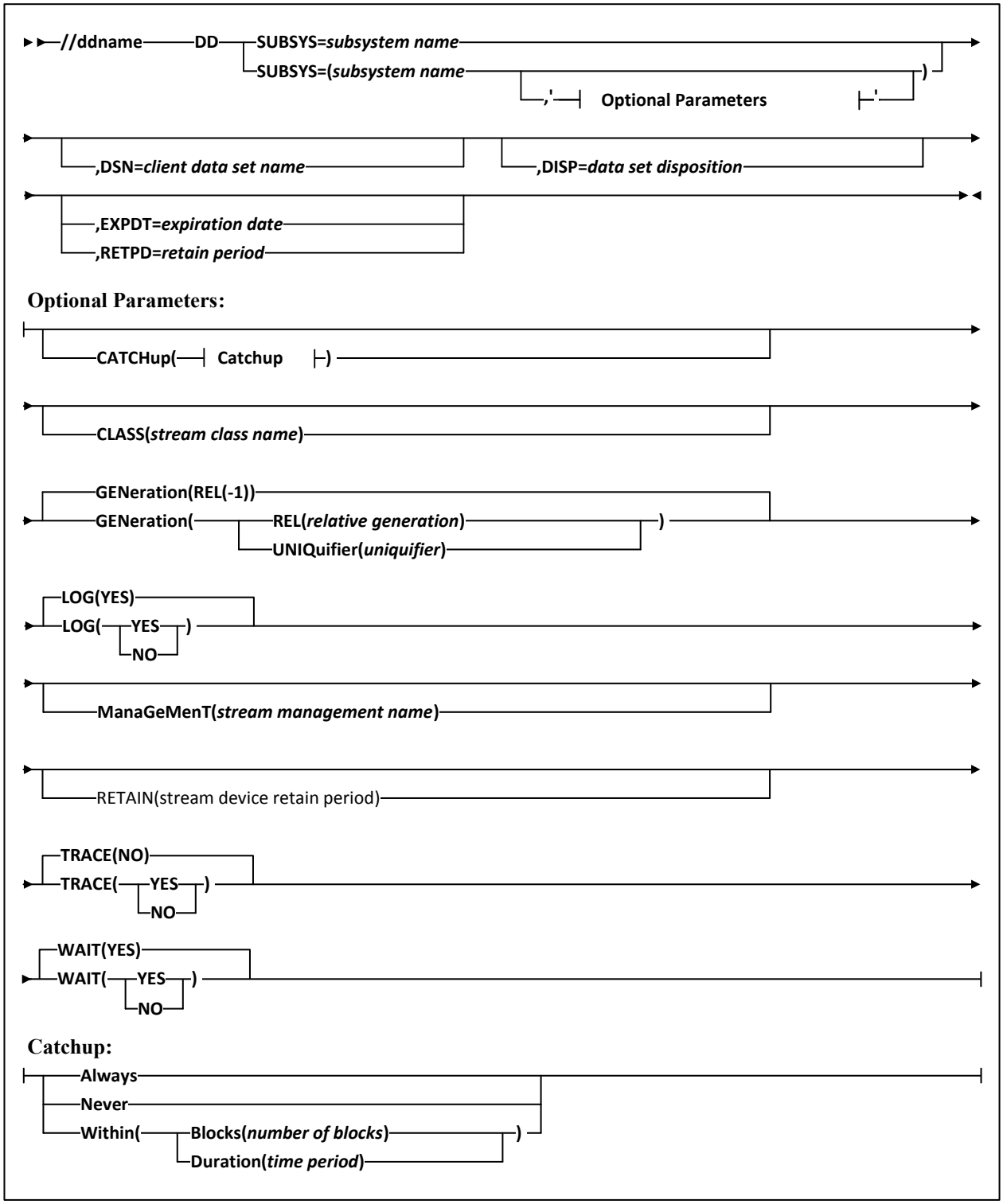


Figure 48. DD SUBSYS parameters

Directive Name

DD SUBSYS is the JCL keyword that initiates the ExHPDM DD SUBSYS parameters. ExHPDM parameters may be specified in any number of DD SUBSYS parameters (up to the JCL limit of 254).

For example:

```
//EGDD1 DD SUBSYS=(SOV,'CLASS(CLASS1)','WAIT(NO)','GEN(REL(-1))')
```

is equivalent to:

```
//EGDD1 DD SUBSYS=(SOV,'CLASS(CLASS1) WAIT(NO) GEN(REL(-1))')
```

Parameters

DSN

DSN is the JCL parameter that specifies the *client data set name* to be used as the connection to ExHPDM. This is either a new data set for write processing or is an existing ExHPDM data set for read processing. Refer to the *MVS JCL Reference* for details on this parameter.

If DSN is not specified, MVS assigns a temporary data set name.

Note: When using FDRABR, then using the DSN is not required. This is because FDRABR assigns its own name to the data set when it is opened.

DISP

data set disposition is the JCL parameter that specifies the client data set disposition. This value is ignored for read requests. ExHPDM only catalogs the client data sets when DISP=(,KEEP,*xx*) or DISP=(,CATLG,*xx*) is specified. Any other value results in the data set not being cataloged. The client data set is cataloged by the ExHPDM server when the client closes the file. If any processing errors occurred during the time that client was connected then the data set is not cataloged. For example, the data set is not cataloged if the client connection is cancelled by an ExHPDM CANCEL command.

All client data sets should be cataloged when using MVS Generation Data Group (GDG) data sets. This is because the ICF catalog identifies the versions of the client data set.

If DISP is not specified, the default is to not catalog the data set.

EXPDT

expiration date is the JCL parameter that specifies the expiry date for the client data set. This value is ignored for read requests. Refer to the *MVS JCL Reference* for details on this parameter.

If this value is 1999/365 (99365), 1999/366 (99366) or a zero day value is specified (for example, 1999/000), the client data set never expires.

If a value is specified in the past (for example, 1998/365) then the client data set expires immediately.

If EXPDT and RETPD are not specified, the ExHPDM ManaGeMentT is used to determine under what circumstances the client data set is expired. This is specified on the DD SUBSYS ManaGeMenT parameter or the CLASS ManaGeMenT parameter. For details on using CLASS ManaGeMenT, see "CLASS Keyword" on page 99. If a ManaGeMenT value cannot be determined for the client data set, the STREAM RETainPerioD is used. For details on using the STREAM RETainPerioD, see "STREAM Keyword" on page 142.

RETPD

retain period is the JCL parameter that specifies how long the client data set is to be kept. This value is ignored for read requests. Refer to the *MVS JCL Reference* for details on this parameter.

If EXPDT and RETPD are not specified, the ExHPDM ManaGeMentT is used to determine under what circumstances the client data set is expired. This is specified on the DD SUBSYS ManaGeMenT parameter or the CLASS ManaGeMenT parameter. For details on using CLASS ManaGeMenT, see "CLASS Keyword" on page 99. If a ManaGeMenT value cannot be determined for the client data set, the STREAM RETainPerioD is used. For details on using the STREAM RETainPerioD, see "STREAM Keyword" on page 142.

CATCHup

CATCHup is used to specify client catchup processing for read requests. Catchup processing occurs when a client is connected to a read stream file after the stream file has started reading data for another client.

The client added to a running read stream may require blocks which are physically prior to the block currently being read. In this case, there are two possibilities:

1. The client may be forced to wait until all currently running clients have finished reading that volume of the stream file.
2. The volume may be rewound to allow the new client to `catch up.' This causes any other clients to be delayed until the new client has read blocks up to the original read position.

Which of these alternatives to choose is specified by the CATCHup parameter. If the new client happens to require blocks beyond the current read position, no special

action is required, since the stream file continues reading in the normal sequence to satisfy the current clients.

The CATCHup parameter may specify one of the following catchup policies:

Always
Always allows a new client to catch up.

Never
Never allow a new client to catch up. The client waits until all other currently reading clients have completed.

Within
allows the client to catchup depending on the following parameter settings:

Blocks
number of blocks specifies that the client is to catchup if the first block of their file is behind the current stream file position by *number of blocks* or less.

Duration
time period specifies that the client is to catchup if it is added to the stream task within this *time period*. *time period* is specified as a time interval as described by "Specifying a Time Interval or Period" on page 161.

CATCHup overrides the similar parameter that can be specified on STREAM as described by "STREAM Keyword" on page 142.

CLASS
stream class name specifies which stream CLASS is to be used. This name can be up to 8 characters in length and must match a CLASS defined in the ExHPDM server parameter file. The *stream class name* is ignored on input (read) connections. This parameter overrides the similar parameter on SELECT as described by "SELECT Keyword" on page 134. If the ExHPDM server is started in disaster recovery (DR) mode, the **CLASS** parameter is not supported.

jGENeration
specifies which ExHPDM generation of the client data set is required. Where there are multiple occurrences of the same client data set name in the ExHPDM database, GEN may be used to obtain a particular version of the data set. This can be selected by either a relative generation number or a uniquifier value. GENeration is ignored for output (write) connections.

Note: ExHPDM generations are not to be confused with MVS GDGs. ExHPDM generations of a client data set are successive copies of those client data sets that have identical names.

REL
The *relative generation* specifies which generation of the client data set by relative position is to be selected. The relative generations may be selected by

counting from the most recent (negative value) or the most ancient (positive value) generation.

If there are *n* generations of a client data set, then REL(–1) is the most recent generation, REL(–2) is the second most recent, and REL(–*n*) is the most ancient generation.

This may also be expressed as REL(1) being the most ancient generation, REL(2) being the second most ancient, and REL(*n*) being the most recent generation.

REL(0) is equivalent to specifying REL(–1).

UNIQuifier

uniquifier specifies a specific instance of a client data set. Each client data set has a unique 16-digit hexadecimal value identifier. This is known as a uniquifier. This value may be displayed using the ADMIN DB LIST commands or may be obtained from the original write client message SOV06204I as written to JESYSMSG.

GENeration(REL(–1)), which specifies the most recent client data set generation, is the default.

LOG

specifies whether output generated as a result of client processing performed by the server is to also be output to the ExHPDM log file. This parameter overrides the similar parameter that can be specified on STREAM as described by "STREAM Keyword" on page 142.

LOG(YES) is the default.

ManaGeMenT

The *stream management name* specifies the MANAGEMENT definition to use for the client data set. This value is ignored for read processing. This parameter overrides the similar parameter that can be specified on SELECT as described by "SELECT Keyword" on page 134, and CLASS as described by "DATABASE Keyword" on page 102.

RETAIN

The *stream device retain period* specifies an overriding retain period to apply to a stream task. The retain period is used to indicate how long a stream task should remain active after all connections to the stream task have completed. The original, and default, stream task retain period is specified by the device RETAIN keyword as described by "DEVICE Keyword" on page 107. The most recent RETAIN specified by a client connection satisfying one of the following conditions becomes the current retain period for the stream task:

- the *stream device retain period* is greater than the current retain period,

- the *stream device retain period* is zero.

Once all connections have completed for a stream task, the stream task enters a retain processing period as indicated by the current retain period. Once this period has elapsed, the stream file and stream device is unallocated and the stream task terminates. The ExHPDM DISPLAY STREAM DETAIL command may be used to determine a stream task in retain processing. For example, the following DISPLAY STREAM DETAIL output shows the write stream is in retain processing for the total period indicated by the Retain Time (1 hour), and will terminate when the Remaining time (currently 31 minutes and 49 seconds) completes:

```
SOV06055I Stream : HSCISTRM ID : 84 Device : 0B80 VOL : 542565
File : OWPL.HSCISTRM.GTR90DH2
Write Flow Rate : 12 KB/sec Blocks : 2297
Status : Waiting
Volumes : Current 1, NNCA 40, Max 50
Elapse Time (HH:MM:SS) : 00:25:47
Retain Time (HH:MM:SS) : 01:00:00 Remaining : 00:31:49
```

If the client connection is for a read, *stream device retain period* becomes the stream read retain period.

If the client connection is for a write, *stream device retain period* becomes the stream write retain period.

RETAIN(0) indicates that no retain processing is to take place. The stream file and stream device are unallocated immediately when no more connections are left to process by the stream task.

If RETAIN is not specified by any client connections, the RETAIN value is used as specified on the DEVICE keyword. See “DEVICE Keyword” on page 107 for further details.

TRACE

specifies if additional trace data is to be generated by client connection. This trace data is captured by an active GTF trace. Tracing must be activated for the ExHPDM server for these trace records to be generated. See “MVS GTF Tracing Services” on page 59 for information on activating server tracing.

TRACE(YES) should only be used when advised by StorageTek Technical Support as its usage can reduce performance for the client data set processing.

TRACE(NO) is the default.

WAIT

specifies if the connection request is to wait (**WAIT**(**YES**)) or is not to wait (**WAIT**(**NO**)) if a connection cannot be satisfied immediately. This parameter overrides the similar parameter that can be specified on STREAM as described by

"STREAM Keyword" on page 142. If the ExHPDM server is started in disaster recovery (DR) mode, the **WAIT(YES)** parameter is not supported.

WAIT(YES) is the default.

DD SUBSYS JCL Parameter Examples

The following are examples of using the DD SUBSYS JCL parameters.

Example 1

The following example specifies a data set to be processed by the ExHPDM server whose subsystem name is SOV.

```
//TAPE1 DD DSN=DBAPROD.IMS001.BACKUP,DISP=(,CATLG),
//                  SUBSYS=SOV
```

Example 2

The following example specifies a data set is to be processed by the ExHPDM server whose subsystem name is SOV. The data set is to be assigned the stream class named DBACLASS and is to be retained for 10 days.

```
//TAPE1 DD DSN=DBAPROD.IMS001.BACKUP,DISP=(,CATLG),
//                  SUBSYS=(SOV,'CLASS(DBACLASS)'),RETPD=10
```

Example 3

The following example specifies a data set is to be processed by the ExHPDM server whose subsystem name is SOV. The data set is to be assigned the stream class named DBACLASS. The connection is not to wait if the ExHPDM stream task resource cannot be acquired. The data set is to never be expired.

```
//TAPE1 DD DSN=DBAPROD.IMS001.BACKUP,EXPDT=1999/366,
//                  SUBSYS=(SOV,'CLASS(DBACLASS) WAIT(NO)')
```

Example 4

The following example specifies that an input data set is to be processed by the ExHPDM server whose subsystem name is SOV. Catchup processing is to occur if the stream task has only been active for 30 seconds. The oldest generation of the client data set is required. Note that the optional parenthesis are not used in the ExHPDM parameters.

Example 5

```
//TAPE1 DD DSN=DBAPROD.IMS001.BACKUP,
//                  SUBSYS=(SOV,'GEN 1 CATCHUP WITHIN DURATION 30secs')
```

The following example specifies that an input data set is to be processed by the ExHPDM server whose subsystem name is SREM. The specific generation of the client data set

OWPL.BACKUP.FDR with the uniquifier value B238A506980D3881 is required. The uniquifier was obtained from the original write client job JESYSMSG from the SOV06204I message as follows:

```
SOV06204I Write client JOB:FDR.DUMP, DSN OWPL.BACKUP.FDR completed:
 uniquifier B238A506980D3881
  started 09MAY1999 23:12:48, ended 09MAY1999 23:14:23, elapsed 00:01:31
 total blocks 1692, I/O time    35.807, delayed    11.716 ( 12.2%)
 avg KB/sec 4625, I/O utilisation  37.3%
```

The DD SUBSYS parameters to obtain this specific client generation is as follows:

```
//TAPE1 DD DSN=OWPL.BACKUP.FDR,
//SUBSYS=(SREM,'GEN UNIQ B238A506980D3881')
```

Example 6

The following example shows the usage of RETAIN to keep a stream task from unallocating a stream file and stream device until all of the job steps have completed. This assumes each step is started within 2 minutes of the previous step running. The final step specifies a RETAIN 0 value which results in the stream task completing immediately the final step completes.

```
//REST1 EXEC PGM=FDR,PARM='R'
//TAPE1 DD DSN=DBAPROD.IMS001.REST1,
//                  SUBSYS=(SREM,'RETAIN 2M')
//DISK1 DD VOL=SER=SPARE1,UNIT=SYSALLDA,DISP=OLD
//SYSPRINT DD SYSOUT=*
//REST2 EXEC PGM=FDR,PARM='R'
//TAPE1 DD DSN=DBAPROD.IMS001.REST2,
//                  SUBSYS=SREM
//DISK1 DD VOL=SER=SPARE2,UNIT=SYSALLDA,DISP=OLD
//SYSPRINT DD SYSOUT=*
//REST3 EXEC PGM=FDR,PARM='R'
//TAPE1 DD DSN=DBAPROD.IMS001.REST3,
//                  SUBSYS=(SREM,'RETAIN 0')
//DISK1 DD VOL=SER=SPARE3,UNIT=SYSALLDA,DISP=OLD
//SYSPRINT DD SYSOUT=*
```

Chapter 11. Changing MVS Parameter Files

Required Changes and Additions to MVS Files

The following changes and additions are required in MVS parameter library files to support ExHPDM.

See the chapter "About This Book" for the syntax conventions used in this chapter. Note that parenthesis are optional in all ExHPDM parameters, however they are shown in the syntax diagrams and examples in the interest of precision.

An example of coding ExHPDM SSI Optional parameters with parenthesis, and the equivalent command coded without parenthesis, and a more readable format, is shown as follows:

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN) INITPARM('PRM(00) TRACE(ON FID(1))')
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN) INITPARM('PRM 00 TRACE ON FID 1')
```

Parenthesis may sometimes be required to ensure that a command is not ambiguous. An ambiguous command may result if a keyword value happened to be the same as a keyword. For example, if a MEMber prefix (used as the parameter member prefix) happened to be called TRACE, then the following command would require parenthesis:

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN) INITPARM('MEMber TRACE')
```

In this example, parenthesis are required because TRACE is a keyword. The minimum parenthesis required are:

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN) INITPARM('MEMber(TRACE)')
```

MVS Subsystem Definition (IEFSSN*xx*)

An MVS subsystem must be defined for ExHPDM in a SYS1.PARMLIB (IEFSSN*xx*) member, where *xx* is the member suffix.

The subsystem definition can be either added to an existing IEFSSN*xx* member or a new IEFSSN*xx* member created. If a new IEFSSN*xx* member is created, the suffix must be added to the IEASYS*xx* member to pick up this specification at IPL. Refer to the

MVS/ESA SP V5 Initialization and Tuning Reference for details about modifying IEFSSN*xx*.

Two definitions of IEFSSN*xx* are possible:

1. Positional parameter specification as shown in the following example. This example shows the positional parameter format used prior to MVS Version 5.2. Although not recommended, positional parameters can also be used on MVS systems at Version 5.2 or higher.

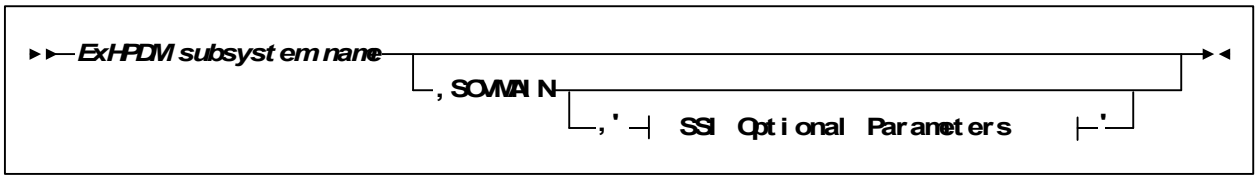


Figure 49. IEFSSNxx Specifications Using SSI Positional Parameters

2. Keyword parameter specifications are shown in the following example. This format is only valid for MVS Version 5.2 and higher.

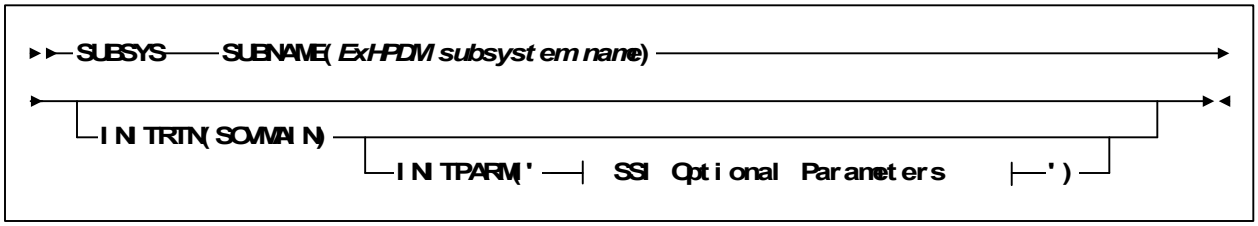


Figure 50. IEFSSNxx Specification Using SSI Keyword Parameters

Note: Note that you cannot mix positional and keyword specifications in the same IEFSSN*xx* member. In addition, the MVS SETSSI ADD command can be used to dynamically create the ExHPDM subsystem without the need of an IPL.

Syntax

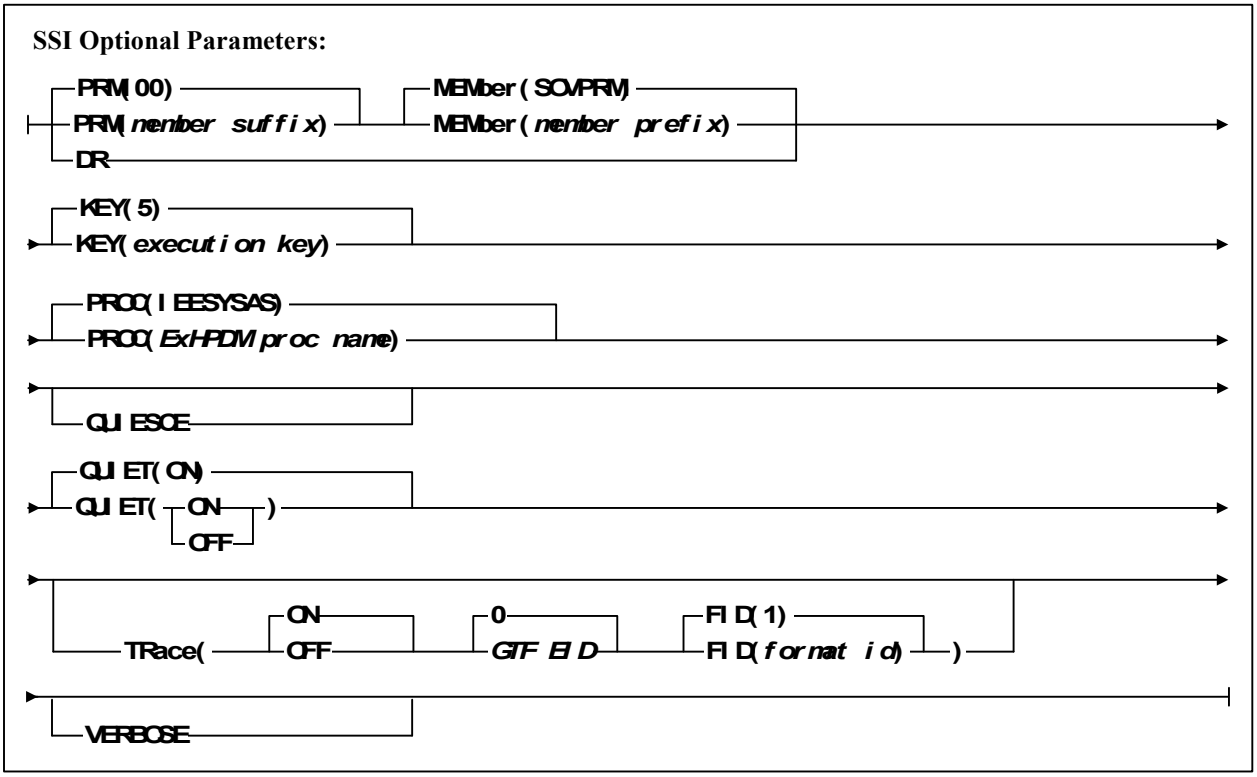


Figure 51. ExHPDM SSI Parameters

MVS Subsystem Definition

ExHPDM subsystem name initiates the MVS subsystem definition for ExHPDM. INITPARM specifies the subsystem parameters when using the keyword form of IEFSSN*xx*. The third positional parameter specifies the subsystem parameters when using the positional form of IEFSSN*xx*. If multiple parameters are supplied, the parameters must be enclosed in single quotes.

Note: When using FDRABR, the default ExHPDM subsystem name of SOV should be used to support the FDRABR DYNTAPE operand. If an ExHPDM subsystem name other than SOV is used, contact Innovation Data Processing Support.

ExHPDM Subsystem Parameters

SOVMAIN
specifies the initialization routine, if it is desired, to start the ExHPDM server at IPL. The ExHPDM load modules must be available in the MVS linklist for SOVMAIN to be specified.

PRM
member suffix specifies the optional two-character suffix for the parameter file member name. The member name is formed using the *member prefix* specified in the

MEMBER parameter. If the STKPARMS DD is a sequential data set (non PDS), the PRM suffix is ignored. If the DR parameter is specified, PRM cannot be specified.

PRM(00) is the default.

MEMBER

member prefix specifies the 1 to 6 character parameter file member name prefix. The member name is formed using the suffix specified in PRM. If STKPARMS DD is a sequential data set (non PDS), the MEMBER prefix is ignored. If the DR parameter is specified, MEMBER cannot be specified.

MEMBER(SOVPRM) is the default.

DR

specifies that the ExHPDM server is to start in disaster recovery mode. In this mode there is no real ExHPDM database. The database is unavailable and usually cannot be reinstated. In disaster recovery mode, ExHPDM startup builds a dummy parameter file internally to simulate the environment for internal processing. The SCANSTReamfile parameter of the ExHPDM Administration Utility is used to locate and restore client data sets. If DR is specified, MEMBER and PRM parameters cannot be specified.

KEY

execution key specifies the PSW key for ExHPDM server in addition to the storage key used by ExHPDM when allocating common storage. Any key between 1 and 15 may be specified. However, problem keys such as KEY(8) should not be specified as this results in common area storage being allocated in this key. The recommendation is to use the default value.

Note: If a KEY is specified that is different to the previously started ExHPDM KEY, a RENEW start is forced.

KEY(5) is the default.

PROC

ExHPDM proc name specifies any valid procedure as setup for the ExHPDM server. An example procedure may be found in the STKSAMP installation samplib member EXHPDM. The values specified in the SSI parameters override the PARM values specified on the proc.

PROC(IEESYSAS) is the default.

QUIESCE

specifies that no access to the ExHPDM database is allowed until a SET DATABASE RESTART operator command is issued. This command is used when restoring a database.

QUIET

specifies whether or not additional internal diagnostic messages are to be output. This parameter should only be used when advised by StorageTek Software Support.

QUIET(ON) is the default.

TRace

turns GTF ON or OFF. This allows GTF tracing to be performed on the ExHPDM initialization processing before the operator command interface is activated. Tracing can be turned ON or OFF by the SET TRace command once the operator command interface is active.

GTF must be active for tracing TYPE=USR events. GTF tracing of ExHPDM events is automatically turned off when ExHPDM is shutdown.

trace id (GTF EID)

trace id specifies that tracing only occurs to GTF where this matches the event identifier (EID) on the USRP GTF keyword for the currently active GTF trace. This may be required to avoid tracing other non-ExHPDM related GTF trace records. A value in the range x'000' to x'3FF' should be specified.

If USRP is not currently in effect for the TYPE=USR GTF trace, tracing always occurs. The selected *trace id* is in effect until a SET **TRACE** command is entered to change the value.

For addition details about using GTF and the USRP keyword, refer to the *MVS/ESA SP V5 Diagnosis: Tools and Service Aids.*

The default *trace id* is 0.

Note: As the values specified on the GTF USRP are specified as hexadecimal values, it may be easier to use hex values for the *trace id*. A hexadecimal value is entered as x'*nnn*'. Where *nnn* matches the GTF EID.

FID

format id specifies the trace formatting appendage to use when formatting the GTF trace entries with IPCS. The *format id* may be any value from, decimal, 0 to 80. The *format id* indicates which AMDUSR*nn* is used by the IPCS GTFTRACE command. Where *nn* is the *format id* and is in the range hexadecimal 00 to hexadecimal 50 (decimal 80). The selected *format id* is in effect until a SET TRACE command is entered to change the value.

The default *format id* is 1.

Note: As the value to determine the AMDUSR*nn* formatting routine is specified as a hexadecimal value, it may be easier to use hex values for the *format id*. A hexadecimal value is entered as x'*nn.*'

VERBOSE

specifies that additional internal diagnostic messages are to be output to the ExHPDM LOGFILE. This parameter should only be used when advised by

StorageTek Software Support. Specifying VERBOSE can result in large amounts of output to be generated to the ExHPDM LOGFILE.

ExHPDM VERBOSE messages are only output when log file messages are being directed to a data set or to SYSOUT. No VERBOSE messages are output when the LOGFILE WTO option has been specified. Refer to “LOGFILE Keyword” on page 117.

MVS Parameter File Examples

The following series of examples show how to define the ExHPDM subsystem using the SSI parameters described. Each example shows the SSI positional parameter method followed by the keyword method.

Example 1

The following example defines the IEFSSN*xx* for subsystem SOV without starting the ExHPDM server at IPL.

```
SOV/*ExHPDM subsystem,positional method*/
```

or

```
SUBSYS SUBNAME(SOV)/*ExHPDM subsystem,keyword method*/
```

Example 2

The following example defines the IEFSSN*xx* for subsystem SOV to use the SOVPRM02 startup file.

```
SOV,SOVMAIN,PRM=02/*ExHPDM subsystem,positional method*/
```

or

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN)
INITPARM('PRM(02)')/*ExHPDM subsystem,keyword method*/
```

Example 3

The following example defines the IEFSSN*xx* for subsystem SOV to use the SOVPRMTS startup file and activate GTF tracing.

```
SOV,SOVMAIN,'PRM(TS) TRACE'/*ExHPDM subsystem,positional method*/
```

or

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN)
INITPARM('PRM(TS) TRACE')/*ExHPDM subsystem,keyword method*/
```

Example 4

The following example defines the IEFSSN*xx* for subsystem SOV to use the SOVPRMTS startup file and activate GTF tracing only if a GTF is currently active for EID (Event ID) 50. The optional parenthesis have not been specified in this example.

```
SOV,SOVMAIN,'PRM TS TRACE ON 50' /*ExHPDM subsystem,positional method*/
```

or

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN)
INITPARM('PRM TS TRACE ON 50')
/*ExHPDM subsystem,keyword method*/
```

Example 5

The following example defines the IEFSSN*xx* for subsystem SOV to use the proc ExHPDM.

```
SOV,SOVMAIN,'PROC(ExHPDM)'/*ExHPDM subsystem,positional method*/
```

or

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN)
```

Example 6

The following example defines IEFSSN*xx* for subsystem SOV to use the EXHPDM02 startup file.

```
SOV,SOVMAIN,MEMBER(EXHPDM) PRM(02) /*ExHPDM subsystem,positional method*/
```

or

```
SUBSYS SUBNAME(SOV) INITRTN(SOVMAIN)
INITPARM('MEMBER(EXHPDM) PRM(02)')
/*ExHPDM subsystem,keyword method*/
```

Example 7

The following example dynamically creates a MVS subsystem using the SETSSI MVS operator command.

```
SETSSI ADD,S=SOV
```

Example 8

The following example dynamically creates a MVS subsystem using the SETSSI MVS operator command. ExHPDM is initialized using the SOVPRMTS startup file and activate GTF tracing.

```
SETSSI ADD,S=SOV,I=SOVMAIN,P='PRM(TS) TRACE'
```

Appendix A: ExHPDM SMF Records

Outputs

The outputs of the ExHPDM SMF feature are SMF records that can be dumped with IFASMFDP or a user–written routine. Different types of data are distinguished by the use of different SMF record subtypes. There are three classes of output records, Audit records, Accounting records, and Performance records as detailed below.

Table 7. SMF Output Records

Class	Subtype	Name
Audit	0	Startup/Shutdown
	1	Client Request
	4	Database Transaction
Accounting	2	Connection Accounting
Performance	3	Stream Task Performance
Disaster Recovery	5	Business Continuance Accounting Record

SMF Mapping Macros

In order to assist in the processing of the subtype records, additional ExHPDM SMF mapping macros have been provided in the STKSAMP library that is created during the installation of ExHPDM. These mapping macros map each of the subtype records and the common header and common job information record. Macros are:

Table 8. SMF Mapping Macros

Macro	Description
SOVSJOBS	ExHPDM Common SMF Job Information Structure
SOVSMFHD	ExHPDM SMF Subtype Macro. Used to generate the assembler DSECTs for all the different subtypes.
SOVSMF0	ExHPDM SMF Subtype 0 record mapping.

Table 8. SMF Mapping Macros

Macro	Description
SOVSMF1	ExHPDM SMF Subtype 1 record mapping.
SOVSMF2	ExHPDM SMF Subtype 2 record mapping.
SOVSMF3	ExHPDM SMF Subtype 3 record mapping.
SOVSMF4	ExHPDM SMF Subtype 4 record mapping.
SOVSMF5	ExHPDM SMF Subtype 5 record mapping.

Common Data Areas

SMF records are composed of an SMF header followed by a subtype record. Subtype records have a common subtype header and a data component.

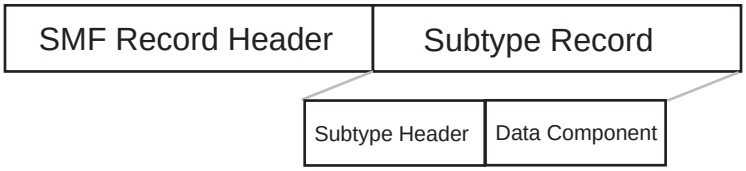


Table 9 on page 292 describes the SMF headers.

Table 9. SMF Headers

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0	0	SOVRLEN	2	binary	Record length. This field and the next field form the record descriptor word (RDW).
2	2	SOVRSEG	2	binary	Segment descriptor. Currently set to zero.
4	4	SOVRFLG	1	binary	Set to 0x1E.
5	5	SOVRRTY	1	binary	Record type.
6	6	SOVRTME	4	binary	Time since midnight.

Table 9. SMF Headers

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
10	A	SOVRDTE	4	packed	Date record was written.
14	E	SOVRSID	4	EBCDIC	System identification.
18	12	SOVRSSI	4	EBCDIC	Subsystem identification.
22	16	SOVRNSR	2		Number of SMF subtype records that follow.

Table 10 on page 293 details the subtype headers. The data component format depends upon the subtype. Table 12 on page 295, Table 13 on page 296, Table 14 on page 297, Table 15 on page 299, and Table 16 on page 301 describe the subtypes and their data components.

Table 10. ExHPDM Subtype Headers

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0	0	SOVHRECL	2	binary	Length of this subtype record including this header.
2	2	SOVHSTNO	2	binary	SMF Subtype number
4	4	SOVHASNM	8	EBCDIC	ExHPDM Server address space name
12	C	SOVHJESN	8	EBCDIC	JES job number or blanks if started SUB=MSTR
20	14	SOVHSUBS	4	EBCDIC	Subsystem name
24	18	SOVHSNAM	8	EBCDIC	System name
32	20	SOVHASID	4	binary	Server ASID
36	24	SOVHSTOD	8	binary	TOD of server startup
44	2C	SOVHTZOS	4	signed	TimeZone OffSet as found in CVTTZ

In addition, some subtype records refer to a common job information structure. The layout of these subtype records is shown in Table 9.

Table 11. Job Information Structure

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0 0 1 2 3	0 0 1 2 3	SOVSJFLG SOVSJFL1 SOVSJFL2 SOVSJFL3 SOVSJFL4	4 1 1 1 1	binary	Flags Word. Job type: X'01' Job is TSO X'02' Job is STC X'04' Job is Batch Reserved Reserved Reserved
4	4	SOVSJNAM	8	EBCDIC	Jobname. TSO: Userid (LOGON Userid) STC: Procname (S PROCNAME) JOB: Jobname (JOB card)
12	C	SOVSSNAM	8	EBCDIC	Job step name. TSO: N/A STC: Identifier (S PROCNAME.IDENTIFIER) JOB: Stepname (EXEC card)
20	14	SOVSJPNM	8	EBCDIC	Job proc step. TSO: N/A STC: Proc step (EXEC card) JOB: Proc step (EXEC card) Only for jobs where a PROC is defined
28	1C	SOVSJNUM	8	EBCDIC	Job id information
36	24	SOV1CNID	4	binary	Console ID if source is console else 0.
40	28	SOV1CNAM	8	EBCDIC	Console name if source is console else blanks
48	30	SOVSJPGM	8	EBCDIC	Job program name.

Audit Records

Audit records are records of requests produced for auditing purposes. A request is any valid ExHPDM request, such as, CONNECT, OPERCMD, ADMIN or VALIDATE. An audit record is also produced for server startup and shutdown.

Startup/Shutdown (Subtype 0)

A startup/shutdown record is produced when the server starts up or shuts down.

Table 12. SMF record subtype 0 - startup/shutdown

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0	0	SOV0FLGS	4	binary	Flags word.
0	0	SOV0TYPE	1		Record type: X'01' Startup X'02' Shutdown
1	1	SOV0FLG1	1		Startup and shutdown options: X'01' Startup was successful X'02' Renew was specified X'04' Shutdown IMMEDIATE was specified X'08' Database quiesce was specified (Startup only) X'10' Disaster Recovery mode (Startup only) X'20' Reserved X'40' Shutdown detected by EOD (ABEND or CANCEL) X'80' Renew was implied (ie. maintenance or new version or key change)
2	2	SOV0FLG2	1		Reserved
3	3	SOV0FLG3	1		Reserved
4	4	SOV0VERS	8	EBCDIC	ExHPMD version information in *v.r.m* format
12	C	SOV0PVER	8	EBCDIC	STK common parser version information in *v.r.m* format
20	14	SOV0PROC	56	Structure	See Table 11 on page 294.
76	4C	SOV0SOVM	6	EBCDIC	SOVPRM*xx* prefix (member name) used at startup. This is combined with the name in SOV0SOVP to format the name of the startup member.
82	52	SOV0SOVP	2	EBCDIC	SOVPRM*xx* suffix used at startup. This is combined with the name in SOV0SOVM to format the name of the startup member.
84	54	SOV0EXEP	128	EBCDIC	Exec parm or SSI parm string used for startup
212	D4	SOV0DBNM	44	EBCDIC	Database VSAM cluster dsn (Blank for DR mode)
256	FF	SOV0JRNL	44	EBCDIC	Journal data set name or blanks if not used

Client Request (Subtype 1)

A request audit record is produced for each client request when the request completes. Note that a CONNECT audit record is written when the connection becomes active in the stream task (or fails to be assigned to any stream task).

A request audit record will also be produced by SMF interval processing if this is activated. In this case, the SOV1FLG1 field will contain X'01'.

Table 13. SMF record subtype 1 - request audit record

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0	0	SOV1FLGS	4	binary	Flags word.
0	0	SOV1FLG1	1		Record type: X'01' Interval record
1	1	SOV1FLG2	1		Request status: X'01' Terminated X'02' Active X'04' Waiting X'08' Waiting for client termination client termination X'10' In use by server X'20' Cancelled
2	2	SOV1FLG3	1		X'40' Request terminated with error
3	3	SOV1FLG4	1		Reserved Reserved
4	4	SOV1RTYP	32	EBCDIC	Record type
36	24	SOV1VID	4	binary	Request identifier
40	28	SOV1ATOD	8	binary	TOD of request acceptance
48	30	SOV1CTOD	8	binary	TOD of request completion
56	38	SOV1RETN	4	binary	Request completion code
60	3C	SOV1REAS	4	binary	Reason code request
64	40	SOV1INFO	4	binary	Request info code
68	44	SOV1CMDL	4	binary	Length of command in SOV1CMDT
72	48	SOV1CMDT	512	EBCDIC	Command text, variable length to a maximum of 512.

Accounting Records

An accounting SMF record is written for each connection upon connection completion.

Connection Accounting Record (Subtype 2)

A connection accounting record contains information about each connection.

Table 14. SMF record subtype 2 - connection accounting record

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0		SOV2FLG2	4	binary	Flags word.
0	0	SOV2FLG1	1		Record type: X'01' Record creation X'02' Write connection request
1	1	SOV2FLG2	1		Status X'01' Connection was successful X'02' Client dataset was cataloged in ICF catalog X'04' Client dataset was closed successfully X'08' Client data was read or written successfully
2	2	SOV2FLG3	1		Reserved
3	3	SOV2FLG4	1		Reserved
4	4	SOV2VID	4	binary	Request identifier
8	8	SOV2DDN	8	EBCDIC	Client DD name, left-justified and blank-padded
16	10	SOV2DSN	44	EBCDIC	Client data set name, left-justified and blank-padded
60	3C	SOV2MEMB	8	EBCDIC	Client member name, left-justified and blank-padded
68	44	SOV2PROC	56	Structure	See Table 11 on page 294.
124	7C	SOV2RETN	4	binary	Client connection completion code
128	80	SOV2REAS	4	binary	Client connection reason code
132	84	SOV2INFO	4	binary	Client connection info code
136	88	SOV2OTOD	8	binary	TOD stamp of connection open processing by the CONNECT processor
144	90	SOV2ATOD	8	binary	TOD stamp of connection becoming active in stream task
152	98	SOV2CTOD	8	binary	TOD stamp of connection close processing
160	A0	SOV2CLAS	8	EBCDIC	Stream class name (write connection only), left-justified and blank-padded
168	A8	SOV2NAME	8	EBCDIC	Stream name, left-justified and blank-padded

Table 14. SMF record subtype 2 - connection accounting record

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
176	B0	SOV2SDSN	44	EBCDIC	Stream file data set name, left-justified and blank- padded
220	DC	SOV2DEVN	8	EBCDIC	Device definition name, left-justified and blank- padded
228	E4	SOV2ESOT	8	EBCDIC	Generic or esoteric of tape device used, left-justified and blank-padded
236	EC	SOV2UNIT	4	binary	MVS device number of tape device used
240	F0	SOV2DEVT	8	EBCDIC	Device type of tape device used, left justified and blank padded.
248	F8	SOV2BSIZ	4	binary	Stream file block size in bytes.
252	FC	SOVUNIQ	8	binary	Client uniquifier
260	104	SOV2BCNT	4	binary	Number of superblocks read or written from/to stream by server on behalf of this client
264	108	SOV2NDLY	4	binary	Number of times a write block was delayed due to an I/O pending completion for another client
268	10C	SOV2IOTM	8	binary	Accumulated I/O time in TOD units
276	114	SOV2DLTM	8	binary	Accumulated delay time in TOD units
284	11E	SOV2WALL	8	binary	Reserved
292	124	SOV2WMNT	8	binary	Accumulated time delayed waiting for stream task tape mount in TOD units
300	12C	SOV2WMTB	8	binary	Accumulated time delayed when a read connection has already passed the first block required for the client (“missed the bus”). Read connections only. Measured in TOD units.
308	134	SOV2WDEL	8	binary	Accumulated time delayed waiting for the client (VSTAM) in TOD units.
316	13C	SOV2RECL	4	binary	Client data set average record length
320	140	SOV2RECN	4	binary	Number of client records
324	144	SOV2MBUF	2048	EBCDIC	Contents of VVRE message buffer, variable length with a maximum of 2K

Performance Records

Whenever a stream task terminates, a performance SMF record is written detailing statistics about the performance characteristics of that stream task.

A performance record will also be produced by SMF interval processing if this is activated. In this case, the SOV3FLG1 field will contain X'01'.

Stream Task Performance (Subtype 3)

A stream task performance (Subtype 3) record is shown below.

Table 15. SMF record subtype 3 - stream task performance record

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0		SOV3FlGS	4	binary	Flags word.
0	0	SOV3FLG1	1		Record type: X'01' Interval record, else termination rec
1	1	SOV3FLG2	1		Stream task type: X'01' Read X'02' Write X'04' Stream file cataloged in ICF catalog X'08' Started by START command X'10' SCANSTREAMFILE stream task
2	2	SOV3FLG3	1		Stream task status flag 1: X'01' Allocating X'02' Mount pending X'04' I/O in progress X'08' Terminating X'10' Cancel pending X'20' Waiting in retain period X'40' Waiting for operator reply X'80' Ready
3	3	SOV3FLG4	1		Stream task status flag 2: X'01' Waiting for work (ie NO work single start command) X'02' Starting, waiting for 'READSYNCH STARTWHEN'
4	4	SOV3STOD	8	binary	TOD stamp of stream task startup
12	C	SOV3RTOD	8	binary	TOD stamp of stream task retain period start, or 0
20	14	SOV3TTOD	8	binary	TOD stamp of stream task termination. May be 0 for an interval record
28	1C	SOV3STID	4	binary	Stream id
32	20	SOV3NAME	8	EBCDIC	Stream name, left-justified and blank-padded
40	28	SOV3SDSN	44	EBCDIC	Stream file data set name, left-justified and blank-padded
84	54	SOV3DEVN	8	EBCDIC	Device definition name, left-justified and blank-padded

Table 15. SMF record subtype 3 - stream task performance record

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
92	5C	SOV3ESOT	8	EBCDIC	Generic or estoteric of tape device allocated, left-justified and blank-padded
100	64	SOV3DEVT	8	EBCDIC	Character tape device media type. May be blanks if open processing not completed or failed
108	6C	SOV3TAPE	8	EBCDIC	Tape media type (STD, 36TRK, EE, SD3A, SD3B, SD3C). May be blanks if open processing not completed or failed
116	74	SOV3UNIT	4	binary	MVS device number of the tape device currently allocated. May be 0 if open processing not completed or failed
120	78	SOV3COMP	4	binary	Tape device compression factor for the entire stream file. May be 0 for interval records
124	7C	SOV3BSIZ	4	binary	Superblock size in bytes
128	80	SOV3BCNT	4	binary	Number of stream file superblocks read or written
132	84	SOV3DROP	4	binary	Number of stream file superblocks dropped. Read stream tasks only
136	88	SOV3VOLC	4	binary	Number of volumes used in stream file
140	8C	SOV3NCON	4	binary	Number of connections processed
144	90	SOV3NBAD	4	binary	Number of connections with error
148	94	SOV3IAVG	4	binary	Stream file device average I/O rate in bytes per second
152	98	SOV3ATIM	4	binary	Device accumulated active time in milliseconds
156	9C	SOV3BTIM	4	binary	Device accumulated block I/O time in milliseconds
160	A0	SOV3WIDL	4	binary	Total time stream task was idle waiting for work with no pending I/O requests in milliseconds

Database Transaction (Subtype 4)

A database audit record is produced for every transaction that updates the ExHPDM database.

Table 16. SMF record subtype 4 - database audit record

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0		SOV4FLGS	4	binary	Flags word.
0	0	SOV4FLG1	1		Database modification type: X'01' Record creation X'02' Record extension X'04' Record deletion
1	1	SOV4FLG2	1		Reserved
2	2	SOV4GLG3	1		Reserved
3	3	SOV4FLG4	1		Reserved
4	4	SOV4VID	4	binary	Reserved
8	8	SOV4PROC	56	Structure	See Table 11 on page 294
64	40	SOV4PLEN	4	binary	Record part data length
68	45	SOV4DKEY	63	EBCDIC	Actual key of data. This is usually the DSN or volser
131	83	SOV4PAD	1	binary	Reserved

Disaster Recovery Records

A disaster recovery SMF record is produced upon the successful completion of an ExHPDM client dataset backup.

Disaster Recovery Audit Record (Subtype 5)

Table 17. SMF record subtype 5 - DR audit

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
0	0	SOV5PROC	56	Structure	See Table 11 on page 294
56	38	SOV5CTOD	8	binary	TOD of request completion (same as SOV1CTOD)
64	40	SOV5RETN	4	binary	Completion code for request (same as SOV1RETN)
68	44	SOV5DSN	44	EBCDIC	Client DD name, left justified and blank-padded (same as SOV2DSN)
112	70	SOV5DDN	8	EBCDIC	Stream file data set name, left-justified and blank-padded (same as SOV2DDN)

Table 17. SMF record subtype 5 - DR audit

Decimal Offset	Hexadecimal Offset	Name	Length	Format	Description
120	78	SOV5SDSN	44	EBCDIC	Stream file data set name, left-justified and blank-padded (same as SOV2SDSN)
164	A4	SOV5STPN	1	binary	Job step number
165	A5	SOV5RESV	1	binary	Reserved
166	A6	SOV5RELV	2	binary	Relative volume number
168	A8	SOV5SBLK	4	binary	Start block number
172	AC	SOV5UNIQ	8	binary	Client uniquilier
180	B4	SOV5VOLC	4	binary	Volume count
184	B8	SOV5FVOL	list	EBCDIC	First volume serial (list of volumes)

Index

Symbols

"ADMIN DATABASE LIST, UPDATE, and DELETE CLIENT selection" on page 212 216

A

ABEND
    ABEND S013 46
    ABEND S614 46
    ABEND U0200 46
ACTIVATE 95
Activate 95
ACTIVATE Keyword
    Examples 97
    Parameters 95
        EnhancedDISPosition 95
        SUBSYSnameINMeSsaGe 97
ADMIN DATABASE BACKUP 249
    Parameters 252
        BACKUP 252
ADMIN DATABASE command 250, 252, 253, 265
ADMIN DATABASE LIST
    Options 237
ADMIN DATABASE LIST, UPDATE, DELETE 216
    Examples 233, 235, 237
    Parameters
        BLocKS (client) 222
        BLocKS (stream) 231
        CReationDaTe (client) 223
        CReationDaTe (stream) 232
        Date Formats 224
        DELETE 218
        EXPIRED 227
        EXPiry Date 235
        FILE 232
        FORCE 218
        GENeration 228
        INCOMPLETE 232
        JOBname 229
        LIST 217
        ListingOPTions 237
        NEWClient EXPIRY 235
        NEWClientOWNER 235
        NEWStream 234
        NEWStreamEXPIRY 235
        NOCATaLoGed 229
        OWNER 230
        PerCenTUTILised 232
        REL 228
        RETainPerioD 235
        SAVEGens 235
        Time Formats 226
        TimeToLive 230
        UPDATE 217
        VOLume 232
        WhenUNCATaloged 235
ADMIN DATABASE RESTORE 248
    Examples 253
    Parameters 252
        APPLYJOURNAL 252
        TOVERSION 252
        VERSION 252
ADMIN DATABASE UPDATE
    Parameters 234
        NEWStream 234
Administration Utility
    SOVADMN 3
Administration utility
    SOVADMN 33
    SYSIN file, directing utilities 34
ADR356E 45
ADR374E 45
ALLOC command and control statement
    syntax 294

B

backup process, ADMIN DATABASE command 250
basic structure, ExHPDM 13
block size, description 2
BLocKS (stream)
    Parameters 231

C

CANCEL Command
- Example 186
- Parameters
  - Client 185
  - CONnection 186
  - REQuest 186
  - Stream 185
- Syntax 185

CANCEL command 184
CLASS 99
CLASS Keyword
- Examples 100
- Parameters
  - ExpectedFlowIncrease 100
  - ManaGeMenT 100
  - STREAM 99
- Syntax 99

CLASS keyword 99
CLASS Keyword, Identifying the performance attributes for the client connection 36
client communications, overview 5
client overview 4
client write processing, overview 4
concepts of ExHPDM 3
CONNECTIONS parameter, load balancing 62
connections, ExHPDM concepts 5
customizing startup parameter file 93

D

data base
- restoring 54

data set profile verification 47
data set, using non-ExHPDM utilities 8
DATABASE 102
DATABASE Keyword
- Examples 105
- Parameters
  - BACKupDSN 103
  - BACKupJouRNal 105
  - JouRNaL 103
  - LIMIT 103
  - QUIESCE 103
  - UNIT 103
  - VOLSER 104

DATABASE keyword 102
DD SUBSYS JCL parameters 273
Deleting Expired Client data sets from the ExHPDM Database 86
DEVICE 107
DEVICE Keyword
- Examples 114
- Parameters
  - ALLOC 108
  - COMPACTion 109
  - ConsiderOFFlineDevices 109
  - DEFAULT 110
  - DEViceLimit 110
  - FLOWLIMIT 110
  - RETAIN 110, 111
  - UNIT 108
  - UNITCouNT 112
- Syntax 107

DEVICE Keyword, Identifying an output device for the stream task 36
DFSMSdss 15, 45
- COPYDUMP 15
- DUMP 15
- RESTORE 15

DFSMSdss and FDR in ExHPDM 2
Disaster Recovery Audit Record 301
disaster recovery mode
- unsupported commands and parameters 256

disaster recovery mode, starting 175
disaster recovery, using ExHPDM 53
DISPLAY 186
DISPLAY Command
- Examples 189
- Parameters
  - Client 187
  - CONnection 188
  - DataBase 188
  - DETail 189
  - DIAGnostics 188
  - ID 188
  - LOG 188
  - OPTions 188
  - REQuests 189
  - Stream 189
  - SUMmary 189
  - VERsion 189

Syntax 187
DISPLAY command 186
dynamic system symbols, definition 93

E

EDM 48
effective flow increase, ExHPDM concepts 5
EnhancedDISPosition 95
ExHPDM
256K Byte block, description 2
administration utility 33
Administration utility, description 33
basic structure 13
CANCEL command 184
concepts 3
DD SUBSYS JCL parameters 273
description 1
diagnostic facilities 57
disaster recovery process, description 53
disaster recovery, overview 34
dump services, use 58
GTF tracing services, use 59
internal tracing facility 58
job termination 45
MVS master trace, use 59
MVS system trace 59
operator commands 183
overview 1
parallelism in function 18, 26
restoring data base 54
SCANSTREAMFILE overview 34
scenarios 71
server, description 34
SHUTDOWN command 204
START command 208
starting in disaster recovery mode 175
starting the server 175
startup parameter file examples 164
startup parameter file, overview 8, 89
trace facilities 57
ExHPDM Concepts 3
ExHPDM concepts
Administration Utility 3
batch jobs SOVDSSU 4
client 4
client communications 5
client write processing 4
connections 5
effective flow increase 5
flow limit 5
server 5
server communications 6
SOVADMN 3
stream 6
stream class 7
stream device 7
stream file 7
stream task 7
subsystem API 6
ExHPDM Database 7
ExHPDM Description 1
ExHPDM Journal Data Set 8
ExHPDM overview, using non-ExHPDM utilities 8
ExHPDM server
identify an output device 42
identify performance attributes 37
load balancing 62
stream and stream task identifying 41
ExHPHM Processing
**SOV* 14
GDG 14
ExpectedFlowIncrease parameter, load balancing 62
Exploiting Parrallelism with ExHPDM
FDR ATTACH 26
MAXTASKS 26

F

FDR 22, 46
Examples 28
FDRABR 27
FDRDSF 26
FDRTCOPY 27
FDRTSEL 27
FDR319 46
flow limit, ExHPDM concepts 5
FLOWLIMIT parameter, load balancing 62

G

GTF tracing services, use 59

H

HFS backup 78

I

Interfacing directly to the TMS
- AutoMedia (ZARA) 49
- CA-1 48
- CONTROL-T 49
- DFSMSrmm 49
- TAPE2000 49
- TLMS 49

internal tracing facility, use 58

J

JCL disposition 96
JCL parameters
- CATCHup 276
- CLASS 277
- DD SUBSYS 273
- DISP 275
- DSN 275
- EXPDT 276
- GENeration 277
- LOG 278
- ManaGeMenT 278
- RETAIN 278
- RETPD 276
- TRACE 279
- WAIT 279
- Within 277

job termination, description 45
journal data set, description 8

K

keyword
- ACTIVATE 95
- CLASS 99
- DATABASE 102
- DEVICE 107
- LKEYINFO 115
- LOGFILE 117
- MANAGEMENT 121
- MONITOR 124
- PREFIX 128
- REQUEST 130
- SAF 133
- SELECT 134
- SMF 140
- STREAM 142
- TMS 155

L

License 9
License Key 9
Linux Backup 79
LKEYINFO 115
LKEYINFO Keyword 115
- Example 116
- Parameters
  - CUSTOMER 115
  - EXPIRY 115
  - KEY 116
  - PRODUCT 115
- parameters 115
- Paramters
  - SITE 115

load balancing for write processing 62
LOGFILE 117
LOGFILE Keyword
- Examples 118
- Parameters
  - DSN 117
  - SYSOUT 118
  - WTO 117
- Syntax 117

LOGFILE keyword 117

M

MANAGEMENT 121
MANAGEMENT Keyword
- Examples 123
- Parameters
  - EQuals99000 122
  - EXPiryDaTe 121
  - RETainPeriod 122

SAVEGens 122
WhenUNCATaloged 122
Syntax 121
ManaGeMenT Keyword, Identifying the stream task for the client connection 36
MAXERRS=1 46
MONITOR 124
MONITOR Keyword
Example 126
Parameters 125
DataBase 125
INTERVAL 125
JouRNaL 125
PERCentageINCrement 125
THRESHold 125
WARNevery 125
Syntax 124
MONITOR keyword 124
MSV time changes 60
MVS dump services, use 58
MVS master trace, use 59
MVS parameter file 283
MVS parameter file changes 283
MVS system trace, use 59
MVS TOD Clock 59

N

NEW,CATLG,CATLG 96
non-ExHPDM utilities, using 8

O

Operation 13
operation overview
overview, operation 13
Operator Commands
CANCEL 184
DISPLAY 186
SET 200
SHUTDOWN 204
START 208
Syntax 183
operator commands
CANCEL 184
DISPLAY 186
ExHPDM 183
SET 200
SHUTDOWN 204
START 208
starting the ExHPDM server 175
operator commands, functions 183
output device, identify 42
overview of functions
client functions 4
overview, ExHPDM 1

P

parallelism with ExHPDM 18, 26
parameter file
start up example 164
parameter file keywords
CLASS 99
DATABASE KEYWORD 102
DEVICE 107
LOGFILE KEYWORD 117
PREFIX KEYWORD 128
REQUEST 130
SAF keyword 133
SELECT keyword 134
STREAM 142
TMS 155
Parameters 95
performance attributes, client connection 37
PREFIX 128
PREFIX Keyword
Examples 128
Parameters
COMMAND 128
MESSAGE 128
Syntax 128
PREFIX keyword 128
PRM 177
processing, ExHPDM 14

R

relationship, stream parameter definitions 43
REQUEST 130
REQUEST Keyword
Examples 131
Parameters
ADMIN 130

CONNECT 130
LOG 131
MAXLIMIT 131
OPERCMD 130
VALIDATE 130
Syntax 130
REQUEST keyword 130
restoring ExHPDM data base 54

S

SAF 47, 133
SAF Keyword
Example 133
Parameters
CLASS 133
Syntax 133
SAF keyword 133
SCANEXPIRE 265
SCANSTReamfile 254
scenarios, ExHPDM 71
SELECT 133, 134
SELECT Keyword
Examples 138
Parameters
CLASS 135, 140
DataSetName 137
EXCLUDE 135
JOBname 137
ManaGeMenT 136
ProGraM 137
SYSname 138
USER 138
Paramters
eXecutionMODE 137
ProcSTEP 137
STEPname 137
Syntax 135, 140
SELECT keyword 107, 134
SELECT Keyword, Identifying a client connection for ExHPDM processing 36
server communications, overview 6
server overview 5
server, ExHPDM
starting 175
server, ExHPDM, description 34
services provided, operator commands 183
SET Command
Examples 203
Parameters
APPEND 201
DataBase 200
DR 177
DSN 201
FID 179
FREE 178
KEY 177
LOGFILE 201
MEMBER 202
NOFREE 178
PRM 177, 201
QUIESCE 177
QUIET 177
RENEW 178
RESTART 200
SSNAME 178
SYSOUT 201
TRace 178, 202
VERBOSE 179
WTO 201
Syntax 200
SET command 200
SET Command Parameters
MEMBER 177
sharing parameter file definitions 92
SHUTDOWN Command
Examples 205
Parameters
IMMEDiate 205
NORMAL 205
Syntax 205
SHUTDOWN command 204
SMF 140
SMF Keyword 140
SMF Mapping Macros 291
SOV06213W 268
SOVADMN 33, 211
Overview 211
SOVDSSU, batch jobs 4
Specifying a Time Interval or Period
Examples 163
Parameters

Days 162
Hours 162
Minutes 162
nn 162
nn.nn 162
nn.nnU 162
nnU.nn 162
Seconds 162
Weeks 162
Years 162
Syntax 161
START Command
Examples 209
Parameters
RETainperiod 208
Stream 208
Syntax 208
START command 208
start up parameter file 8, 89
customizing 93
start up parameter file, example 164
startup parameter file, format 90
static system symbols, definition 93
STREAM 142
stream class, ExHPDM concepts 7
stream device, ExHPDM concepts 7
stream file, ExHPDM concepts 7
stream identifying, client connection 41
STREAM Keyword
Examples 151
Parameters
CATaLoG 145
CATCHup 147
CONCURRENT 145
CONNECTIONS 145
DEVice 145
DSN 144
ElapsedTIME 147
EXPiryDaTe 146
LOG 146
NO 149
OPERreply 147
READ 145
ReadSYNCh 146
RETainPerioD 148
RETRYperiod 148
SEParateby 149
STARTwhen 147
stream name 144
TOTalCONNections 147
WAIT 148
WRITE 145
Syntax 143
STREAM keyword 142
STREAM Keyword, Identifying the stream task for the client connection 36
stream parameter definitions, relationship 43
stream task identifying, client connection 41
stream task, ExHPDM concepts 7
stream. ExHPDM concepts 6
streaming verification 47
SUBSYSnameINMeSsaGe 97
subsystem API, concept 6
subsystem API, overview 6
Subtype 5 Record 301
SYSIN file, directing utilities 34
system authorization 46
system security 46
system symbols, dynamic 93
system symbols, parameter file definitions 92
system symbols, static 93
system verification 46

T

Technical Overview
subsystem API 6
TMS 155
TMS Keyword
Examples 159
Parameters
CA1 156
CONTROLT 156
DataManagerNAME 158
NONE 155
RMM 156
SCRatchVOLumeRouTiNe 156
SSNAME 158
TAPE2000 156
TLMS 156
USER 156
ZARA 156

Syntax 155
TMS keyword 155
trace facilities 57
tracing
GTF tracing services 59
internal tracing facility 58
MVS master trace 59
MVS system trace 59

U

Using AutoMedia (ZARA) with ExHPDM 52
Using CA-1 with ExHPDM 50
Using CONTROL-T with ExHPDM 50
Using DFSMSrmm with ExHPDM 50
Using TAPE2000 with ExHPDM 51
Using TLMS with ExHPDM 52

V

VERBOSE, use 58

W

write client connection, performance attributes 37